电脑学习机上机指导系列丛书之五

LOGO 语言

于 春 李观勇 编著

電子工業出版社

内 容 简 介

本书是电脑学习机上机指导系列丛书的第五册。重点介绍了电脑绘图王国——LOGO 语言。

LOGO 语言是面向绘图的计算机高级语言，它的主要特点是。命令容易扩充，支持递归调用，简单的几条指令就可以绘出出乎意料的复杂图形。因些，它是绘图效率最高的计算机语言。

本书共分“LOGO 与学习机、立即执行方式绘图、过程入门、海龟作品、递归过程、系统管理、数值计算、字表处理、应用程序集锦”等十二章，较全面、系统地阐述了 LOGO 语言的特点、结构和编程方法。

LOGO 语言

于 春　李观勇　编著

责任编辑　施玉新

特约编辑　李　明

*

电子工业出版社出版

北京市海淀区万寿路 173 信箱(100036)

电子工业出版社发行　各地新华书店经销

北京大中印刷厂印刷

*

开本：787×1092 毫米 1/32　印张：9.375 字数：210 千字：

1996 年 1 月第一版　1996 年 1 月第一次印刷

印数：5000 册　　定价：9.80 元

ISBN 7-5053-3072-1/TP·1072

前　言

家庭电视游戏业的发展，突飞猛进、一日千里。随着游戏机的普及，各种以任天堂系列 8 位游戏机为主机、配备键盘而构成的“普及型家庭电脑学习机”也迅速推出。目前生产厂已有三十余家，产品品种近百个。较有名的品牌有：中山小霸王、敬东金字塔、北京裕兴以及深、广一带的金钥匙、和声、科达、科特、美猴王、宏源、智源等，产品档次也已拉开了低、中、高三个级别。有关资料表明，目前我国电脑学习机的拥有量已逾 800 万台。

现在，学习机的功能日趋完善，实用性越来越强。除小霸王还继续维持其寓教于乐的游戏特点以外，各大生产厂家都向提高学习机的实用性进军。其工作系统已由从日本引进时的单一的 F BASIC(整数)语言发展到中文 BASIC(浮点)语言、LOGO 语言，汇编语言、中文编辑、文字处理等系统。就硬件来说，94年初裕兴率先在其 YX4型机上推出标准微机101键盘，随后和声、科达、金钥匙等立即推出了仿101键盘的产品，小霸王、金字塔也于10月份推出了仿101键盘的486B 和9588产品；金字塔在94年率先推出了外接中华机140K 容量的5.25英寸软盘驱动器之后，95年新春伊始裕兴又于推出了与微机完全兼容的3.5英寸标准软驱。从应用软件来看，发展更加迅速，已推出了小学、中学乃至高中的各年级辅助教学软件，特别是裕兴推出的仿“金山北大 WPS6.0”的“裕兴 WPS 文字处理系统”，不仅操作与微机一致，而且文件可与微机相互交流。从而，使学习机的档次又升一级而迈进了电脑阶层，使它变成了名符其实的“家庭电脑”(Family Computer)，完全

可以充当现代家庭管理、办公自动化管理的廉价微机。

为了使电脑初学者能够迅速入门，进而登堂入室。我们依据循序渐进的原则，编写了该丛书。本套丛书分：

操作入门
文字处理
游戏 BASIC 语言
中文 BASIC 语言
LOGO **语言**
汇编语言
电脑作曲
电脑绘图
BASIC **语言程序集**
硬件与维修

共十册。

本书是丛书的第五册。重点介绍了电脑绘图王国——LOGO 语言。

LOGO 语言是美国麻省人工智能实验室专为青少年和电脑初学者设计的计算机语言。它以孩子们十分喜欢的搭积木拼图方式，通过控制“海龟”在屏幕上绘图来学习计算机编程，训练学生们的逻辑思维和创造才能。

LOGO 语言是面向绘图的计算机高级语言，它的主要特点是：命令容易扩充、支持递归调用，简单的几条指令就可以绘出出乎意料的复杂图形。因此，它是绘图效率最高的计算机语言。

本书共分“LOGO 与学习机、立即执行方式绘图、过程入门、海龟作品、递归过程、系统管理、数值计算、字表处理、应用程序集锦”等十二章，较全面、系统地阐述了 LOGO 语言的特

点、结构和编程方法。全书内容翔实、语言简练、示例生动，定能指导你灵活的驾驭海龟在图画的海洋中遨游。

本书由苏子栋教授审定。在编写过程中，陈建宇、韩玉文、于勇、于腾潇等同志作了大量的工作，在此一并表示感谢。

编　者

1995年3月于北京

丛书学习说明

本丛书是面向电脑初学者的科学普及读物。丛书的编写结构安排，从初学者最易接受的角度及学习兴趣循序渐进、逐步深入，引导初学者在不觉中步入电脑的殿堂。

丛书的前五册为基础篇，重点介绍了电脑基础知识、电脑学习机的选购常识以及键盘、软盘驱动器、打印机微机外设的基础知识；常用汉字输入法和中文文字处理操作方法；F BASIC、中文 BASIC、LOGO 三种计算机语言的指令及编程方法。

丛书的后五册为提高篇，重点讨论了6527CPU 工作系统的特点、汇编语言的指令及编程方法；音乐板、PLAY 指令、机器语言三种电脑作曲方法；键控、BASIC 指令、机器语言三种绘图方法；精选了60个 F BASIC、中文 BASIC 语言程序；最后讨论了电脑学习机的硬件结构原理和基本维修方法。

整套丛书内容连贯，但又独立成章。十本书读者可依序学习，也可在学习了丛书之一《操作入门》后，根据自己的兴趣和爱好挑选学习。如需要进行文字处理的用户，只要再学习丛书之二就可胜任；爱好音乐的朋友可继续学习丛书之七，就可令电脑演奏音乐，若要进一步提高品位，还须学习丛书之五，使用机器语言编程才能演奏美妙的旋律；电脑绘图爱好者，则要在丛书之三、之八上下功夫；许多朋友在学习了游戏 BASIC 语言后，想自己动手编写任天堂游戏程序及各种实用程序，那么丛书之九给提供了各种类型的程序范例供你参考。

本丛书的撰写宗旨就是辅导读者掌握电脑的基础知识，有深造要求的朋友，请继续参阅任天堂游戏理论书籍《F BA-

目　录

第一章　电脑学习机综述 …… (1)

第一节　电脑学习机的基本结构 …… (1)
第二节　计算机语言与 LOGO 语言 …… (4)

第二章　LOGO 与学习机 …… (7)

第一节　启动 LOGO 系统 …… (7)
第二节　LOGO 的屏幕格式 …… (8)
第三节　LOGO 的指令 …… (10)
第四节　怎样学习 LOGO …… (11)

第三章　立即执行方式绘图 …… (13)

第一节　海龟作图的基本命令 …… (14)
第二节　简单直线型海龟图 …… (24)
第三节　重复命令 …… (27)
第四节　海龟隐身术 …… (34)

第四章　过程入门 …… (36)

第一节　过程和过程编辑 …… (37)
第二节　过程的嵌套 …… (43)
第三节　出错信息 …… (52)
第四节　过程的编辑 …… (54)

第五章　带参数的过程 …… (61)

第一节　参数及带参数的过程 …… (62)

第二节　形式参数和实在参数 …… (69)
第三节　圆和圆弧 …… (70)
第四节　圆和圆弧过程的应用 …… (75)
第五节　多角星 …… (81)
第六节　海龟的坐标与定向 …… (88)
第七节　彩色世界 …… (99)

第六章　海龟作品 …… (104)

第一节　直接命令绘图 …… (104)
第二节　编写过程绘图 …… (109)

第七章　递归过程 …… (115)

第一节　递归与递归过程 …… (116)
第二节　条件转移 …… (124)
第三节　无条件转移 …… (139)
第四节　中间递归 …… (140)
第五节　多重递归 …… (145)
第六节　直接递归和间接递归 …… (153)
第七节　递归的深入讨论 …… (154)

第八章　系统管理 …… (166)

第一节　工作空间管理 …… (166)
第二节　文件管理 …… (169)
第三节　图形及程序打印 …… (172)

第九章　数值计算 …… (174)

第一节　LOGO 中的数及变量 …… (174)
第二节　算术运算和表达式 …… (176)

第三节　输出 …… (179)
第四节　赋值 …… (185)
第五节　标准函数 …… (191)
第六节　应用题举例 …… (202)

第十章　字表处理 …… (211)

第一节　字与字运算 …… (212)
第二节　表与表运算 …… (220)
第三节　键盘扫描 …… (226)
第四节　测试与检测 …… (229)
第五节　变量的深入讨论 …… (235)
第六节　字和表的应用 …… (240)

第十一章　结构化程序设计 …… (246)

第一节　结构化程序设计方法 …… (246)
第二节　程序设计标准 …… (248)
第三节　程序设计举例 …… (250)

第十二章　应用程序集锦 …… (258)

第一节　键控绘图程序 …… (258)
第二节　工具类程序 …… (260)
第三节　加法练习程序 …… (263)
第四节　复杂图形绘制程序 …… (267)

附录一　错误信息一览表 …… (279)
附录二　LOGO 指令一览表 …… (280)
附录三　画笔颜色代码对照表 …… (284)
附录四　ASCII 码表 …… (285)

第一章　电脑学习机综述

电子游戏机是微电子技术和计算机技术高度发展的产物。实际上它就是一部简化了的微型计算机。

电子游戏机进入家庭，深受孩子们的喜爱，也吸引着许多大人参与。不仅如此，专们设计的游戏机还可以用于军事训练、模拟空战演习、汽车驾驶员培训等等。

随着电子游戏机的普及，电脑学习机也走进了千家万户。利用家庭已经普及的电视机、录音机、再配备键盘和工作系统（学习卡）就构成了“五脏具全”的简易微机——电脑学习机。

第一节　电脑学习机的基本结构

同微型计算机一样，电脑学习机也是由硬件系统和软件系统构成的。

一、硬件系统

电脑学习机的硬件结构，由主存储器，运算器，控制器，以及输入/输出设备，外存储器等部件构成。通常把控制器和运算器称为中央处理器，其组成示意如图 1—1 所示。

通常，人们将主存储器、运算器、控制器通称为学习机的主机；而把输入/输出设备和外存储器等总称为学习机的外部设备。

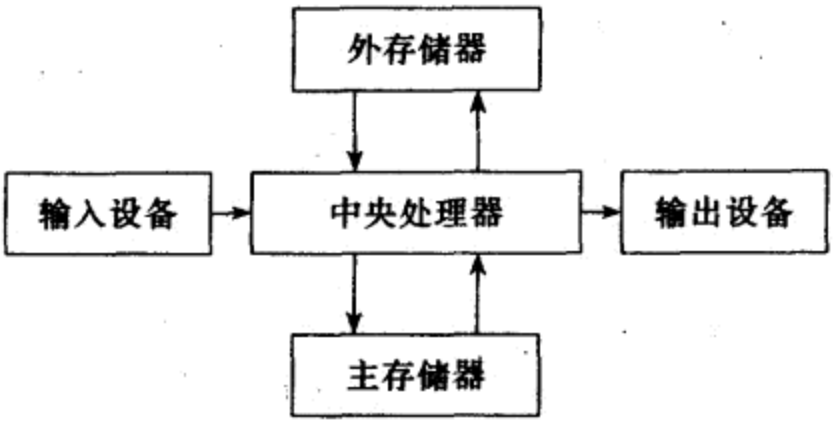


图 1—1　电脑学习机硬件配置示意图

控制器——是学习机的“神经中枢”，它发出控制命令并按一定的顺序执行各种操作。比如什么时候取数，从哪里取数，数送到哪里；进行什么运算，运算的结果送到哪里等等。

运算器——它的任务是进行运算。

主存储器——用来存放系统的常驻信息以及当前被中央处理器所处理的信息和数据。它还可以存放待编辑或运行的程序，保存和记忆原始数据、运算步骤及中间结果等。主存储器的空间是有限的，只能存放当前所处理的信息和数据，但存取速度较快。停机后，主存储器中的信息将全部消失。

外存储器——用来存储当前暂时不处理的信息和数据。常用磁盘、磁带等作为存储介质。外存储器可将信息与数据永久保存。外存储器的容量较大，但存取速度较慢。在外存储器中，信息一般是以文件形式存放的。

输入输出设备——用于向学习机输入数据、信息或由学习机输出结果。输入设备一般指键盘；输出设备一般指电视机、显示器、打印机等。

二、软件系统

电脑学习机与其它电子设备的本质区别是：模拟人脑的功能对信息进行加工处理。但模拟人脑的能力，并不是学习机本身固有的。实际上，学习机的模拟能力是由人们编制的程序控制实现的(这些程序一般固化在学习卡中)。没有程序的支持，学习机将是一堆无用的摆设。它只有在执行程序才能显示出它强大的能力。由此可见，学习机所做的一切，还是由人控制的。

学习机的工作过程，就是执行程序的过程。比其它电子设备的高明之处，在于它能存储这些程序。学习机存储的程序越多，作用就越大(因此，学习卡就有了 2 兆、4 兆、8 兆之分)。这些各种各样的、控制学习机运行的程序的总称，叫做学习机的软件系统。

软件系统是学习机不可缺少的重要部分。它包括系统软件和应用软件两大类。

系统软件——主要指操作系统。F BASIC、中文 BASIC、LOGO 语言就是操作系统的组成部分。没有系统软件的学习机是无法工作的。好的软件才能充分发挥硬件的作用和能力。因此，一个学习机所包含的操作系统种类越多，功能就越强。

应用软件——包括通用应用软件和专用应用软件，如英汉词典、辅助教学、打字游戏等，是为完成某种专门数据处理、计算等而编制的应用程序。

硬件、软件两大系统组成一套完整的学习机，二者缺一不可。

第二节　计算机语言与LOGO语言

一、语言

语言是人类之间相互交流思想、传递感情、交换信息的工具。要使计算机按照人的意图工作，就必须使它懂得人的想法，接受人向它发出的命令和传达的信息。因此，产生了沟通人和计算机的桥梁——计算机语言。

由于目前的计算机还不能识别人的语言和动作，因此只有通过计算机语言让它知道你要做什么，它才能为您服务。

计算机语言，是一种能够直接或间接地被计算机接受的、由有限的字符、按一定规则构成的语句、数据或命令的集合。通过计算机语言，编写特定的程序，就能让计算机按照人们规定的方式去工作。

二、低级语言

早期的编程语言是“机器语言”，即一串“0”，“1”的符号。机器语言识别困难、易于出错、使用不便。于是，人们用一些简单而又形象的符号来代替每一条具体的指令，形成了汇编语言。人们编程时使用汇编语言编写，计算机执行时再将汇编语言翻译成机器语言，这样给编程带来了一定的方便。

汇编语言和机器语言相比较，向前迈进了一大步，但是它与人类习惯使用的自然语言相比，差距还是很大的，与数学语言的距离也比较大，加上它与机器语言是相对应的，所以也是一种面向机器的语言，通用性差、难以推广。一般将机器语言和汇编语言称为低级语言。

三、高级语言

随着计算机的不断发展，使用计算机的人越来越多，人们渴望能有一种面向人的解题过程、类似数学语言、接近自然语言的计算机语言。经过人们长期不懈地探索和研究，第一个高级语言——FORTRAN 语言终于在1956年问世。

高级语言又称“程序设计语言”或“算法语言”。高级语言的产生和应用，对计算机的普及、推广与应用起到了重大的推动作用，是计算机发展史上的重要里程碑。高级语言本身也不断扩充和完善。到现在为止，全世界投入使用的高级语言已达数百余种。其中比较著名的有：BASIC 、Pascal、LOGO 、FORTRAN 、dBASE 等等。

高级语言的最大优点是脱离了计算机的内部逻辑结构，面向用户，面向人的解题过程，类似于人们熟悉的数学语言和自然语言。因此，高级语言具有易学、易用、直观、简洁，通用性强、移植性好等优点。

四、LOGO 语言

LOGO 语言是一种面向孩子、侧重绘图的计算机高级语言，由美国麻省理工学院人工智能实验室的西摩·佩帕特教授研制成功。

LOGO 语言一问世，立即得到孩子们的垂青。他们抛弃了最喜爱的玩具，沉浸在神奇无比的创造中流连忘返。也就因为它的引导而造就了一批科技人才，据说世界上许多著名的专家、学者就是在 LOGO 的启蒙下投身于科学事业的。

LOGO 语言跳出了传统的编程模式，摒弃了屏幕显示坐标的概念，以日常生活对话形式的指令、以搭积木拼图形的程

序结构，控制形象的画笔——海龟，绘制出各种结构复杂、神奇绝妙、出乎意料的美丽图案。操作计算机如同作游戏，使孩子们在迷人的图形设计中学会了计算机编程技能，不知不觉中锻炼了逻辑思维能力，自己培养了自己的创造才能。

LOGO 语言还具有先进的模块化程序结构，不仅使孩子们从小养成了良好的模块化程序设计习惯，而且还培养了孩子们的结构化思维方法。

第二章　LOGO与学习机

目前，LOGO语言在国外仍然十分流行，仍在培养着千千万万个“神童、博士”，而在我国则随着中华学习机的淘汰而消声匿迹。电脑学习机的兴起，又为LOGO语言注入了新的活力，又使它重现昔日的魅力，使我们的小天使又睹到LOGO语言的风采。当然电脑学习机也因增添了LOGO语言而倍增身价。

LOGO语言是以点作图的工作系统，它的显示原理与F BASIC、中文BASIC截然不同，必须采取特殊的软件处理才能实现。因此，目前电脑学习机虽然种类繁多，但能直接支持LOGO工作系统的只有“金字塔”、“裕兴”两种机型。

由于裕兴“新世纪”的LOGO语言除支持磁带存储程序外，还支持3.5英寸磁盘存储程序，使用灵活、操作方便。我们仅以该机型为例进行介绍，但所有的示例程序和编程方法对裕兴、金字塔各种版本的LOGO语言系统都是适用的。

第一节　启动LOGO系统

在系统主菜单下，移动手指光标到LOGO语言一栏，按回车键就进入LOGO工作系统，映入眼帘的是图2－1所示的画面。

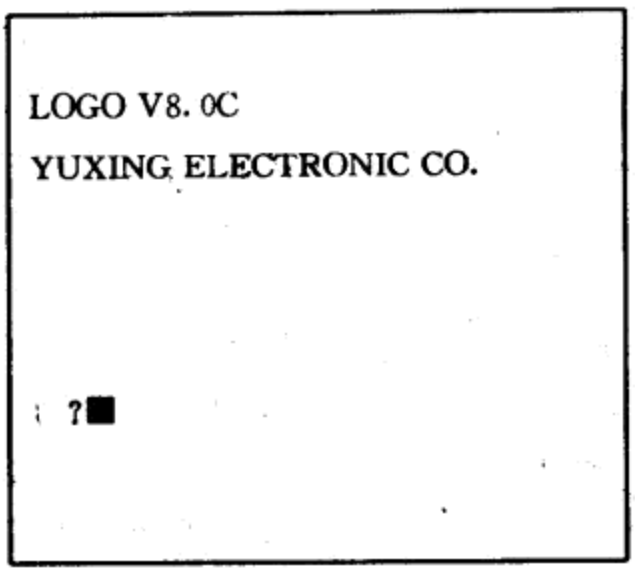


图2－1　LOGO 工作画面

这时的屏幕方式正处于纯文本工作状态。在左下方的“?”是 LOGO 工作状态提示符，“?”后的闪烁光标“■”表示系统已经做好准备，等待接收你键入的命令。

第二节　LOGO 的屏幕格式

一、LOGO 语言的屏幕格式

LOGO 语言有三种屏幕格式：

纯文本屏幕格式

纯图形屏幕格式

图文混合屏幕格式

下面简单说明这三种屏幕格式的用途。

1. **纯文本屏幕格式**　又称“文本屏”。它是显示程序及执行程序后所得文字结果时采用的一种显示方式。文本屏以字

符为计算单位，全屏共有25行，每行显示28个字符（显示分辨率为28列×25行）。它主要用于人机对话，不能显示图形。

2. **纯图形屏幕格式**　又称“图形屏”。它是显示程序及程序执行后所得图形结果时采用的一种显示方式。图形屏以点为计算单位，宽256点、高240点（显示分辨率为256列×240行）。是体现 LOGO 优秀绘画能力的重要窗口。

3. **图文混合屏幕格式**　又称“混合屏”。它是 LOGO 使用对话式绘图时使用的一种显示方式。在这种方式下，屏幕被分割成上下两部分，上部为图形区，有20行字符高（200点），用来以点为单位显示图形；下部为文本区，有5行字符高，用于显示程序或程序执行所得的文字结果（其中只有上面的4行有效）。文本区将遮盖其下面所显示的图形，因此，要想看到完整的图形必须切换到图形屏。

二、屏幕方式的切换

LOGO 语言的三种屏幕显示方式可以任意切换，切换的方法是：

按 F1键或键入 TEXTSCREEN 命令，进入文本屏
按 F2键或键入 SPLITSCREEN 命令，进入混合屏
按 F3键或键入 FULLSCREEN 命令，进入图形屏

三、海龟

当进入图形屏或混合屏时，屏幕的中间将显示一个尖头朝上的白色三角形（如图2－2所示）。它就是我们的主角，大家都叫它“海龟”（取在浩瀚的图画海洋中，任意遨游之意）。它实际上相当于一支画笔，屏幕就相当于一张画纸。我们可通过键入的 LOGO 命令或执行编写的 LOGO 程序，去控制海龟的位

置、方向、着色，令它在屏幕移动，留下各种各样的“龟迹”，组成丰富多彩的图画。

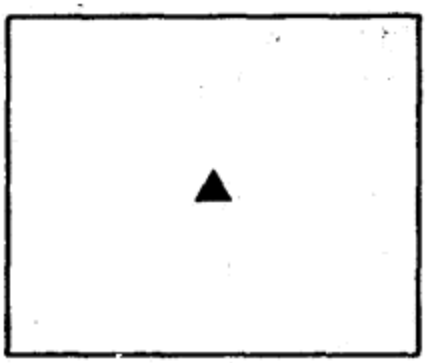


图2—2　海龟

第三节　LOGO 的指令

一、LOGO 指令的种类

LOGO 的指令按功能可分为五大类：

用于绘图的指令

用于过程的指令

用于计算的指令

用于字表处理的指令

用于输入输出操作的指令

这些指令大部分由英文大写字母组成。LOGO 语言虽然指令较多，但都来源于英文单词或英文单词的缩写，所以特别方便记忆。

另外，LOGO 中还有一部分由复合热键组成的命令。

二、LOGO 指令的介绍方式

LOGO 指令的词一般较简洁，为了便于读者记忆，在每条指令的介绍中，均说明该指令词的来源和对应的中文含义。介绍次序依次为：格式、类型、简写或热键、功用、注释、说明、示例(没有项将省略)。

格式——说明命令的使用方式。

类型——说明指令的类型是命令、语句或函数。一般命令和语句可以单独使用，函数则只能用在命令或语句中。

简写或热键——说明该指令的简写形式或具有相同功能的按键。

功用——说明该指令的功能作用。

注释——说明该指令的英文来源和对应汉语的含义，对含有参数的指令，说明参数的使用方式，取值范围等。

说明——对命令的使用进行补充叙述，指出有关的注意事项。

示例——举出指令的使用实例。

第四节　怎样学习 LOGO

一、建立轻松愉快的环境

实践证明，轻松愉快的情绪有利于启发思考、开拓思维、调动智力，而紧张的情绪则恰恰相反。

初坐在电脑键盘前面的孩子，一般都较紧张，要热情地鼓励他们大胆而友好地和电脑交谈，鼓励他们在轻松的探索中去认识、熟悉 LOGO；让他们在通俗而直观的 LOGO 命令面

前指挥那可爱的小“海龟”作图；使得学习、使用电脑成为一件充满兴趣的事。

二、全家人共同参与

在海龟的帮助下，学习电脑简直就是一种娱乐和游戏。人们可以比较轻松愉快地了解计算机的知识、它的工作、解题过程，学习程序设计的基本概念和方法。可以说 LOGO 是一种罕见的计算机高级语言，它可以使祖孙三代产生共同的兴趣，交流共同的语言。同时，它又具有现代程序语言中许多先进的特性(递归、自扩充、结构化品格、易用性等)，为学习其它计算机高级言语打下良好的基础。

三、天才来自勤奋

可爱的小海龟必将会成为你的好伙伴。但是小海龟喜欢交往勤奋、聪明的朋友，不愿意和那些懒惰、愚蠢的人打交道。如果你摸不清海龟的脾气，它将会不听从你的指挥。如果你想驯服海龟，那么就要真正地精通 LOGO。否则它将会从自由的遨游变为趴在地上一动也不动的大懒虫。

要想精通 LOGO，必须付出辛勤的汗水和劳动。只要肯下功夫，小海龟才会伴随你轻松地步入电脑世界之门。这个门虽然对许多人至今仍然神秘，但它正是联系电脑外行和内行之间的关系纽带。

第三章　立即执行方式绘图

LOGO 语言最简单的绘图方法，就是从键盘上敲入一条命令按一次回车键，海龟绘一段图形；再敲一条命令再绘一段图形。这种绘图方式称为立即执行方式，其最大特点是：简单、直观，随时输入命令随时看到结果，特别容易掌握。

本章将结合立即执行方式绘图，讨论如下指令：

混合屏幕格式设置指令	SPLITSCREEN
海龟前进指令	FD
海龟后退指令	BK
海龟右转向指令	RT
海龟左转向指令	LT
抬笔指令	PU
落笔指令	PD
回家指令	HOME
清屏指令	CS
清屏回家指令	DRAW
清除文本屏指令	CLEARTEXT
图文清除指令	ND
重复执行指令	REPEAT
绘图纵横比设置指令	.ASPECT
藏龟指令	HT
显龟指令	ST

第一节　海龟作图的基本命令

一、作图准备

日常绘画要先准备好画笔、画纸和颜色。电脑绘画也必须做好这些准备。

立即绘图方式一般使用混合屏，这样既能看到输入的命令又能看到绘出的图形。进入混合屏的指令是按 F2键或键入 SPLITSCREEN 命令。

格式：SPLITSCREEN

类型：命令

热键：F2

功用：进入混合屏幕格式。

注释：SPLIT 在英文中为“分裂、割裂”的意思，SCREEN 在英文中为“屏幕”的意思。两词合在一起表示“分裂屏幕”，即把屏幕分为两部分。

进入 LOGO 系统，按 F2键或键入

?SPLITSCREEN ↙

命令后，屏幕闪一下，原来屏幕上部的内容被清除，海龟显示在屏幕中央（头向正上方），提示符“?■”显示在屏幕倒数第四行的最左端。这就是说，最下四行用来显示文字信息（键入的命令或系统输出的信息），屏幕上部则用于海龟画图。

海龟作图“状态”包含两个含义：海龟在屏幕上的位置和海龟的方向。刚进入混合屏时，海龟位于屏幕中心的位置，称为海龟的“母位”（也称海龟的“家”）。海龟尖头所指的方向则是海龟前进的方向（绘图的方向）。

二、前进指令 FD

格式:FD　参数

类型:命令、语句

功用:令海龟沿着尖头的方向前进规定的距离。

注释:FD 来源于"ForwarD",是"前进"的意思。

参数表示海龟前进的步数(海龟的一步相当屏幕上的一个点)。从屏幕的中心开始,向上最多走120步,向下最多走119步,向左最多走128步、向右最多走127步,否则就会走出屏幕。这条命令控制海龟前进所给定的步数。

参数的取值范围为0～999999999,也可以取变量或表达式的形式。

说明:命令与参数之间必须要加空格,否则计算机将输出错误信息。

示例:键入

?FD　50 ↙

回车后,将看到海龟向前爬了一段距离,在它身后留下一条直线(图3－1A)。如果再键入

?FD　50 ↙

回车,海龟将会继续向前爬行一段距离,同样留下直线(图3－1B)。再键入

?FD　30 ↙

回车,现在,海龟将冲出屏幕,从屏幕的低部冒了出来(图3－1C)。这说明海龟向上前进的步数已经出了范围。

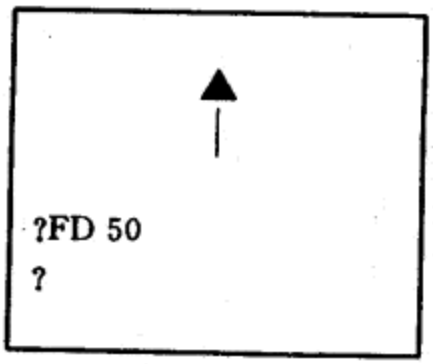


图　3－1A

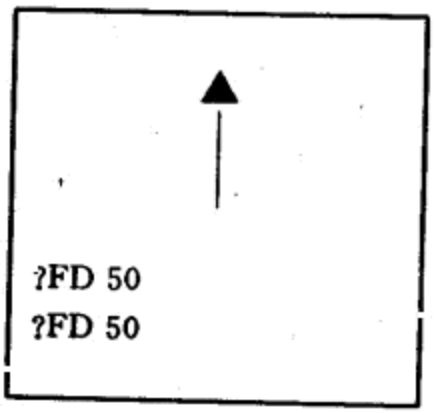


图　3－1B

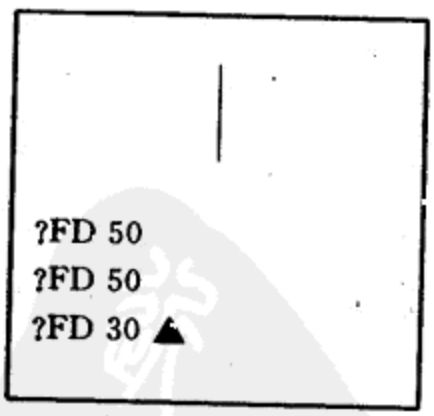


图　3－1C

三、后退指令 BK

格式：BK　参数

类型：命令、语句

功用：令海龟沿着尖头的方向后退指定的距离。

注释:BK 来源于"BacK",是"后退"的意思。

参数表示后退的步数,取值范围为0～999999999,也可以取变量或表达式的形式。

说明:命令与参数之间必须要加空格。

示例:键入

?BK　40 ↙

回车后,海龟后退40步。

下面留一个例题,请读者自己上机实验,注意每输入一条命令回车后,海龟所处的位置。

FD　40 ↙

FD　60 ↙

BK　50 ↙

FD　80 ↙

BK　130 ↙

FD　239 ↙

四、转向指令

前面已经提道,海龟作图有两个状态因素,其中一个就是控制海龟的方向。如果海龟不能转向,总是在一条直线上前进、后退,这样对作图来说就没有什么意义了。所以 LOGO 语言又提供了两条使海龟拐弯的命令。

1. **右转向命令** RT

格式:RT　参数

类型:命令、语句

功用:令海龟向右转动指定的角度。

注释:RT 来源于"Right Turn"。是"向右转弯"的意思。

参数表示角度值,单位为"度"。可以取数值常数、变量或

表达式的形式，取值范围为0～999999999。参数为正时海龟向右转；参数为负时海龟向左转。如果参数是算术表达式，系统会自动对算术表达式的值取整。

说明：海龟以尖头向上为0°，所有的顺时针转向都叫做右转向。

示例：

```
?FD  50 ↙
?RT  90 ↙
?FD  50 ↙
```

2. **左转向命令** LT(LEFT)

格式：LT　参数

类型：命令、语句

功用：令海龟向左转动指定的角度。

注释：LT 来源于“Left Turn”。是“向左转弯”的意思。

参数的解释参见右转向指令 RT。

现在已学习了四条指令，下面让我们上机，一边观察、一边操作，看看怎样画出一个正方形。

例3－1　画出边长为50步的正方形。

```
?FD  50 ↙
      (海龟前进50步，画出一条直线)见图3－2A。
?RT  90 ↙          (右转90度，海龟头向正右方
?FD  50 ↙          (海龟前进50步，见图3－2B
?RT  90 ↙          (右转90度，海龟头向下方
?FD  50 ↙          (海龟前进50步，见图3－2C
?RT  90 ↙          (右转90度，海龟头向正左方
?FD  50 ↙          (海龟前进50步，见图3－2D
```

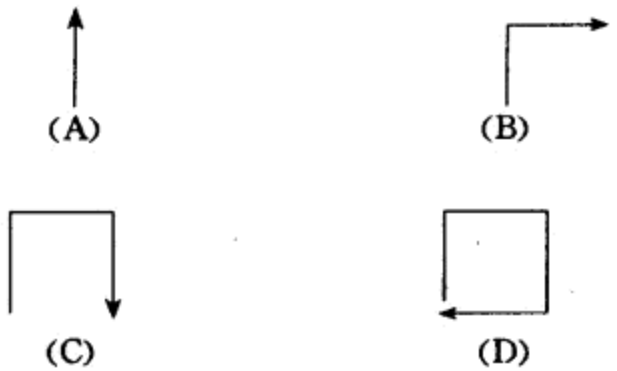


图3—2 海龟画正方形

上面的命令序列，不必每输入一条命令打一次回车键，可以在一行中打入几条命令，各命令之间用空格隔开。当一行中输入多个命令时，这些命令将依次连续执行。下面的命令行和上面的几行命令效果是一样的。

?FD 50 RT 90 FD 50 RT 90 FD
50 RT 90 FD 50 ↙

上面所画的正方形偏在屏幕的一边，如果要把它画在屏幕中心，或者说要使正方形对角线的交点和屏幕的中心重合，就要用到抬笔和落笔命令了。

五、抬笔和落笔指令

1. **抬笔指令** PU

格式：PU

类型：命令、语句

功用：抬起画笔，使海龟在移动过程中不留下痕迹，即不作图。

注释：PU 来源于“Pen Up”，是“笔抬起”的意思。

说明：执行该命令后，海龟所有的移动都不留痕迹，只有在执行落笔命令后才可继续作图。

示例：键入

?UP ↙

?FD 50 ↙

仅海龟向前移动了50步，没有画出线条。

2. 落笔命令 PD(PENDOWN)

格式：PD

类型：命令、语句

功用：落下画笔，使海龟在移动过程中绘图。

注释：PD 来源于“Pen Down”，是“笔落下”的意思。

说明：电脑在进入 LOGO 系统后，自动设置成落笔状态。

示例：

例3－2 画一条如图3－3所示的长为15，间隔为5的纵向虚线。

请键入：

```
?FD  15        (画线)
?PU            (提笔)
?FD  5         (空走)
?PD            (落笔)
?FD  15        (画线)
?PU            (提笔)
?FD  5         (空走)
?PD            (落笔)
?FD  15        (画线)
```

图3－3

掌握了抬笔、落笔命令后，就可以将例3－1的正方形画在屏幕中央了（只要在命令序列前面插入抬笔、落笔命令就可以了）。命令如下：

?PU LT 90 FD 25 LT 90 FD 25
LT 180 PD↙ （令海龟从母位移到左下角）
?FD 50 RT 90 FD 50 RT 90 FD
50 RT 90 FD 50↙
现在画出的正方形就在屏幕的正中了。

六、清屏和海龟回家指令

学习了以上几个基本绘图命令后，就可以练习绘制一些简单的图形，如三角形、五角星等。不一会就把屏幕画得乱七八糟，分不清哪些图形是现在画的。因此，必须把屏幕擦干净才能继续绘图。

1. 清屏回家指令 DRAW

格式：DRAW

类型：命令、语句

功用：置混合屏，令海龟回到母位。

注释：DRAW 在英文中是“画画、绘图”的意思。在此表示令海龟处于绘图准备状态。

说明：在画一个新图形之前，首先要清屏，并令海龟回到母位，以便决定绘图坐标。

执行 DRAW 命令不清除内存中的语句。

2. 清屏指令 CS

格式：CS

类型：命令、语句

功用：清屏，不改变海龟的状态。

注释：CS 来源于“Clear Screen”，是“清除屏幕”的意思。

说明：执行此命令后，屏幕图形全部消失，但海龟不回到母位，保持原来的位置状态。

示例：

```
?DRAW                    (清屏，令海龟回到母位
?FD  30                  (向上画一条线段
?LT  60  FD  50(向左上方画一线段
?CS                      (清除所画图形，不改变海龟状态图
```

在执行 CS 命令时可看到海龟并没有回到母位。

3. **回家指令** HOME

格式：HOME

类型：命令、语句

功用：使海龟回到母位。

注释：HOME 在英文中为“家”的意思。在此表示令海龟回到母位。

说明：HOME 不同于 DRAW，该命令虽使海龟回到母位，但没有清屏的作用。另外，当执行 HOME 命令时，海龟从当前位置回到母位这段距离的痕迹，将根据抬笔、落笔、置色的状态来决定是否画出。

示例：

例3－3　海龟回家的演示

```
?DRAW                    (清屏并使海龟回到母位
?FD 60 RT 120 FD 60      (海龟作图、见图3－4A
?HOME                    (海龟回家、见图3－4B
```

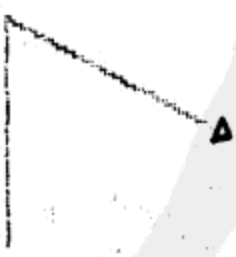


图3－4A

图3－5B

执行 HOME 命令后，由于当前海龟处于落笔绘图状态，故海龟回家中又多画出了一条斜线，使原图形组成了一个三角形。

比较 DRAW、CS、HOME 三个命令，它们的各自特点是：

CS 命令，清屏幕，海龟不回到母位；

HOME 命令，使海龟回到母位，但不清屏幕；

DRAW 命令，使海龟回到母位，并且清屏幕。

实际上 DRAW 命令相当于 CS、HOME 两个命令的功能。

4. **图文清除命令** ND

格式：ND

类型：命令、语句

功用：退出海龟作图状态，清屏；进入文本屏，光标移到屏幕的左下角。

注释：ND 来源于“No Draw”，是“不绘图”的意思。在此取结束绘图，返回文本屏之意。

说明：执行此命令时，屏幕上的图形和文字全部消失，光标移到屏幕左下角。

示例：在例3－3的后面再打入 ND 命令，将会看到屏幕全部被清除，并且光标移到屏幕的左下角，等待你重新输入命令。

5. **清文本屏命令** CLEARTEXT

格式：CLEARTEXT

类型：命令

功用：清除文本屏，光标移到屏幕的左下角。

注释：CLEARTEXT 在英文中为“清除文本”的意思。

说明：此命令只能在纯文本屏幕下使用。

以上讨论了关于屏幕清除和海龟回到母位的几条控制命

令，各命令的功能各有所长。在使用时要根据实际情况选择使用。比如想把屏幕上的图形擦掉，重新作图，就要用 CS 命令；如果想叫海龟回到屏幕中央不清屏，就要用 HOME 命令；如果既要海龟回到屏幕中央又要清屏，就用 DRAW 命令或者用前面提到的屏幕切换法；如果想退出作图状态，并清除所有的图形和文字，要用 ND 命令；如果处在纯文本屏幕时，想清除屏幕上的文字，可以用 CLEARTEXT 命令等等。

以上介绍了一些基本绘图命令，这些命令又可分成两类：一类是无参数命令，如 DRAW、PD、CLEARTEXT 等，这些命令使用时，只写命令名；另一类是带参数命令，如 FD、RT、BK 等，这些命令使用时，除命令名外，后面还必须跟有相应的参数值。

第二节　简单直线型海龟图

一、简单图形绘制实例

使用以上学习的命令，已可以绘出许多有趣的图形。本节将介绍几个简单的直线型海龟图实例供读者练习。

例3－4　画图3－5所示的边长为50步的正六边形。

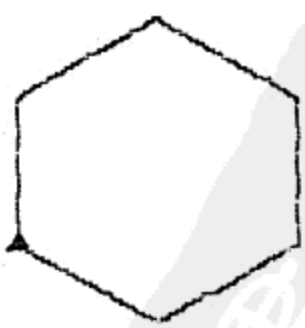


图3－5　正六边形

绘图命令为：

```
?DRAW
?FD  50  RT  60  FD  50  RT  60
?FD  50  RT  60  FD  50  RT  60
?FD  50  RT  60  FD  50  RT  60
```

例3－5　画如图3－6所示的边长为60步的正五角星

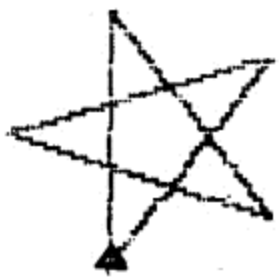


图3－6　五角星

绘图命令为：

```
?DRAW
?FD  60  RT  144  FD  60  RT  144
?FD  60  RT  144  FD  60  RT  144
?FD  60  RT  144
```

例3－6 画一个如图3－7所示的领奖台，每个台阶的宽、高均为30步。

图3－7　领奖台

绘图命令为：

```
?DRAW
?FD  30  RT  90  FD  30  LT  90
?FD  30  RT  90  FD  30  RT  90
?FD  30  LT  90  FD  30  RT  90
?FD  30  RT  90  FD  90
```

例3—7 绘制如图3—8所示的两个正方形(要求画在屏幕的中心)。

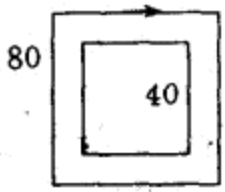


图3—8 双正方形

绘图命令为:

```
?  PU
?  FD  20
?  PD
?  RT  90  FD  20  RT  90  FD  40  RT  90
?  FD  40  RT  90  FD  40  RT  90  FD  20
?  PU
?  LT  90  FD  20
?  PD
?  RT  90  FD  40  RT  90  FD  80  RT  90
?  FD  80  RT  90  FD  80  RT  90  FD  40
```

二、设置屏幕纵横比例

细心观察,以上所绘的图形,纵向较扁、横向较长。这是纵横比例失真所造成的。为了弥补这一不足,必须在绘图前设置

屏幕纵横比。设置屏幕纵横比的指令为“.ASPECT”。其格式如下：

格式：.ASPECT　参数

类型：命令、语句

功用：调整绘图的纵横比例，使绘图符合人们的视觉习惯。

注释：.ASPECT 在英文中为“比例”的意思。前面加一标识符点号“.”，表示设置图形纵横比例。

参数为横向坐标与纵向坐标的比值。当参数取1.1或1.2时所绘图形在视觉上较为适宜。参数可以省略，省略参数时的默认屏幕纵横比为1。

注意：这个命令以小数点开头，小数点不能省略。

上面例子的图形，基本上是重复使用前进和转向命令所完成的，绘图过程中完全回避了各顶点的坐标值。这使得那些没有“坐标”概念的人也能学习 LOGO 语言。

既能学习计算机语言编程，又把所要求的数学知识水平降到很低，这也是 LOGO 语言的一个特点。

第三节　重复命令

在前面介绍的绘图实例中，很多命令是重复的。如在例3—7画正方形的命令中，仅右转90°就使用了8次之多。为了简化命令输入，提高编程效率，LOGO 语言提供了重复执行命令 REPEAT。

一、重复命令的一般格式

格式：REPEAT　参数　[命令串]

类型：命令、语句

功用：使[命令串]代表的程序段重复执行若干次，重复执行的次数取决于 REPEAT 后的参数。

注释：REPEAT 在英文中为“重复”的意思，在此表示重复执行中括号中的命令。

参数的值必须大于等于0，可以取数值常数、变量或表达式。

注意：REPEAT、参数、括号之间必须用空格隔开、参数最好是大于1的正整数，如果不是整数，计算机将自动取整，如果小于1则无意义。

在立即执行方式下，回车表示命令输入的结束，执行命令的开始，电脑接受到回车信号，就去执行刚键入的命令。因此，当命令较长时，一行输入不下，可继续输入，电脑会自动换行，在下一行接着显示键入的命令。

使用重复命令画正多边形最方便，让我们先来讨论正三角形的画法。

绘制图3－9所示的正三角形（边长为70步），步骤如下：

从母位开始，令海龟前进70步，画出第一边；然后令海龟转向120°（正三角形的三个内角是60°，由于海龟转的角是外角，所以应取补角），绘第二边。因此，把上述命令重复执行三次就可画出正三角形。命令为：

```
REPEAT 3 [FD 70 RT 120]
```

执行该命令，立即画出正三角形。

画正多边形的关键是计算海龟的转向角度，对于正 n 边形其转向角度 a 可用如下公式求出：

$$a=360/n$$

有了这个公式，绘制多边形命令的设计十分简单。“a”就是海

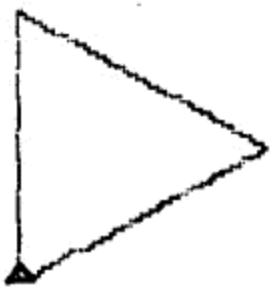


图3－9　正三角形

龟的转向角度、海龟前进的步数就是边长，重复执行命令的参数就是边数。

例3－8　使用重复命令分别画出边长为40步的正三角形、正方形、正五边形、正六边形和正七边形。

用下列五行命令就可以画出图3－10所示的五个正多边形：

```
REPEAT  3  [FD  40 RT  120]
REPEAT  4  [FD  40 RT  90]
REPEAT  5  [FD  40 RT  72]
REPEAT  6  [FD  40 RT  60]
REPEAT  7  [FD  40 RT  360/7]
```

仔细观察上面五条命令，可以发现这样一个规律，绘制多边形的重复命令中：

重复次数×转动角度≡360°

几何学告诉我们，正多边形所有内角之和等于 360°。正多边形的各个内角都相等。所以每个内角恰好是360°/边数。按照这一规律，画 n 边形，转动的角度应该是360°/n。

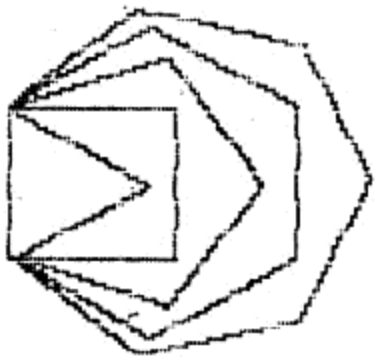


图3－10

二、用重复命令画圆

LOGO 所画的曲线实际上都是由许多细分的直线组成的，也就是说曲线的微观组合是折线。比如画一个圆圈，可以反复地执行下述命令：

FD 1 RT 1

显然，画圆只要反复执行上述命令行360次就可以了。请观察下面命令的运行结果：

REPEAT 360 [FD 1 RT 1]

执行这个命令，海龟就扭扭捏捏、曲曲折折地画出一个圆圈。若把这个命令与画正多边形的命令相比较，会发现这个圆实际上就是一个正360边形。

绘图结束后，你会发现画出的圆太扁，有点象椭圆。这可以用“.ASPECT”命令进行补偿，参数可根据电视机的特点在1.2～1.4之间选取。

总之，LOGO 所绘出的曲线都是这样用许多短线段连接而成的。如画四分之一圆的一段弧，可用命令：

REPEAT 90 [FD 1 RT 1]

实现。

例3－9　画四个不同直径的圆内相切。

绘图命令如下：

```
DRAW
.ASPECT 1.4
REPEAT  360  [FD  1  RT  1]
REPEAT  180  [FD  1  RT  2]
REPEAT  120  [FD  1  RT  3]
REPEAT  90  [FD  1  RT  4]
```

执行上述四条画圆命令后，绘出的图形如图3－11所示。

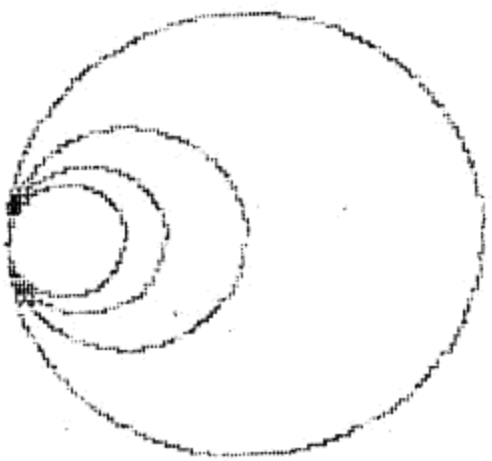


图3－11　内相切的四个圆形

三、重复命令的嵌套

重复命令可以嵌套，也就是说在一个重复执行的命令中，可以包含另外一个重复命令。这种情况称为重复命令的嵌套。使用重复命令嵌套可以用简单图形的绘图命令绘制出妙趣横生的图形。下面列出几例供欣赏。

例3－10　由正六边形转动绘制的图形。

键入命令：

REPEAT 6 [REPEAT 6 [FD 25 RT 60] RT 60]

绘出的图形如图3－12所示。

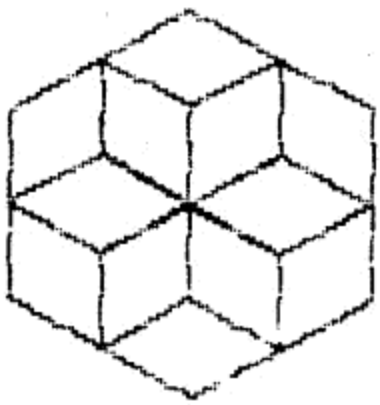


图3－12 正六边形转动绘制的演绎图形

例3－11 由正方形转动绘制的图形。

键入命令：

REPEAT 6 [REPEAT 4 [FD 40 RT 90]

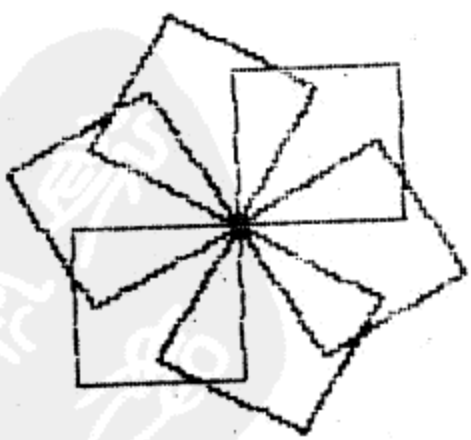


图3－13 正方形转动绘制的演绎图形

RT 60]绘出的图形如图3－13所示。

例3－12 由正十二边形转动绘制的图形。

键入命令：

REPEAT 12 [REPEAT 12 [FD 15 RT 30] RT 30]绘出的图形如图3－14所示。

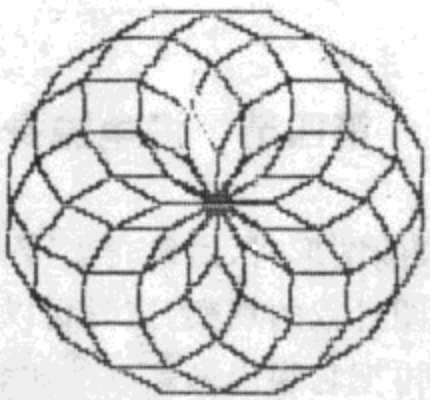


图3－14 正12边形转动绘制的演绎图形

例3－13 由圆转动绘制的图形。

键入命令：

REPEAT 18 [REPEAT 120 [FD 1 RT 3] RT 20]绘出的图形如图3－15所示。

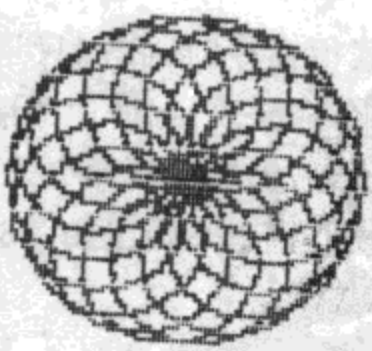


图3－15 圆形转动绘制的演绎图形

通过上面几例可以明显看出，使用重复命令可以简化操作。

希望读者细心体会上述例子中 REPEAT 命令的运用技巧，加强 REPEAT 命令的实际应用练习，熟练掌握重复命令。

注意：在使用嵌套方式时，千万不要忘记命令与命令之间的空格。

第四节　海龟隐身术

当画较复杂的图形时，海龟转来转去，有时挡住了我们的视线，不利于绘图过程的观察。这时可以使用藏龟命令先将海龟藏起，需要时再把它唤出来。

一、藏龟指令 HT

格式：HT

类型：命令、语句

功用：令海龟在屏幕上消失。

注释：HT 来源于“HideTurtle”，是把“海龟藏起来”的意思。

说明：在执行藏龟命令时，我们只是看不见海龟，其实海龟的指向和位置状态都没改变。海龟的隐藏，并不意味着不能画图，只要是落笔状态，仍然可以命令隐藏的海龟画图。

例3－14　在藏龟状态下画出如图3－16所示的图形。

键入命令：DRAW

```
HT
FD 30
LT 90
```

```
FD 15
RT 120
FD 50
RT 120
FD 50
RT 120
FD 15
LT 90
FD 30
RT 90
FD 20
```

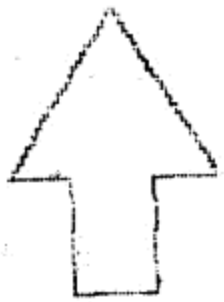


图3—16　箭头

二、显龟指令 ST

藏龟后，将看不到海龟的当前状态，往往迷失绘图的方向。因此，有时又必须把海龟唤出来。唤出海龟的方法是执行显龟命令“ST”。

格式：ST

类型：命令、语句

功用：令隐藏的海龟显示出来。

注释：ST 来源于“Show Turtle”，是“显示海龟”的意思。

掌握了这两条命令，你可以在不需要海龟登场时把它隐藏起来，需要时再把它召唤出来。

第四章　过程入门

前面所讨论的绘图方式属交互式绘图。在这种方式下，编写绘图命令使用的是 LOGO 语言的原始指令，画一个图形必须键入一组命令。由于输入命令按回车键后，电脑会立即执行绘图。因此，又称为立即执行方式绘图。

立即执行方式的优点是“输入即所得，绘图直观”，但也存在很大的缺点。主要缺点是电脑等待时间长，而且输入的命令只能执行一次，第二次绘图仍须再输入一遍命令。也就是说输入的命令不能保存，使用效率低。

实际绘图中（特别是在画一个复杂的图形时），绘图命令往往要进行多次的修改和调式。如果用立即执行方式，多数时间将耗费在命令的输入过程中。因此，人们希望象 BASIC 那样，把编写的绘图命令保存起来，可以进行多次修改和运行，这就要用到 LOGO 语言的“过程和过程编辑”。

本章将结合“过程”的定义和编辑，介绍：

过程定义指令	TO
过程结束指令	END
确认过程编辑热键	Ctrl—C
放弃过程编辑热键	Ctrl—G
语句分行热键	Enter
前删除热键	Esc
后删除热键	Ctrl—D
快速后删除热键	Ctrl—K

光标移至语句行首热键	Ctrl—A
光标移至语句行尾热键	Ctrl—E
光标上移一屏热键	Ctrl—B
光标下移一屏热键	Ctrl—F
内容卷至屏首热键	Ctrl—L

第一节　过程和过程编辑

“过程”(procedure)是LOGO语言最基本、最重要的语法部分，是LOGO程序的最基本模块。LOGO的许多特点和优点都将通过“过程”体现出来。

一、过程的概念

“过程”在LOGO中，既是一段独立的绘图命令集合，又可作为LOGO命令直接使用。也就是说，可以使用已有的LOGO命令编制新的LOGO命令，用以扩展LOGO的命令集。被编制的新命令一经确认，就能像LOGO原始命令那样直接使用。

“过程”相当于BASIC中的程序，它由指令或命令序列组成。“过程”要在编辑状态下形成，称为“过程”定义。定义过程前要先起一个名字，称为“过程名”，过程名的作用是方便过程的执行或调用。“过程”一经定义即存入内存，它既可以保存在内存中供编辑、修改或随时调用，还可以存在磁盘或磁带中长期保存，供以后使用和交流。

二、定义过程指令 TO

格式：TO 过程名

类型：命令

功用：定义一个过程。

注释：TO 在英文中为“给予”的意思，在此表示“形成一个过程”。

过程名是用户按自己的需要给过程指定的名字，它可以由若干个以英文字母开头的字母、数字组成，如 WORLD、CHINA、C3 等，同时也可以插入一些除运算符以外的其他符号。一般情况下，总是给过程起个能表示过程所完成的某种操作或执行结果的名字，以起到标识符的作用。例如画圆的过程，可以使用汉语拼音“YUAN”来作过程名，也可以选用英语单词“CIRCLE”。这样可以帮助我们记忆，也便于以后使用时查找。但是新的过程名字不能同 LOGO 固有的命令重名，也不能和已经定义的其它过程重名。

三、过程的定义

假如我们欲定义一个画正方形的过程，取名为“ZFX”，在文本屏或混合屏下，键入

? TO ZFX ↙

这时，屏幕上的图形和文字将全部被清除，显示如图 4－1 所示的画面。

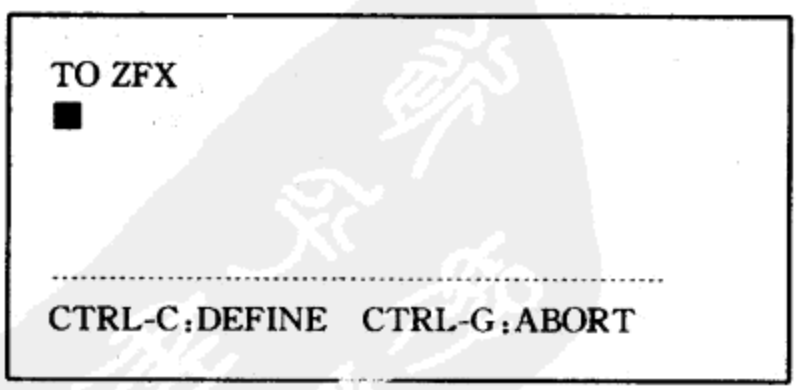


图 4－1 过程编辑画面

图 4－1 的画面就是过程编辑屏幕，第一行是刚打入的过程命令，表示开始定义过程“ZFX”；最下面的一行是操作提示，“Ctrl－C”的含义是：确认定义的过程，退出编辑屏幕；“Ctrl－G”的含义是：废除定义的过程，退出编辑屏幕（注：“Ctrl－C”是复合热键，操作方法是按住“Ctrl”键不放，再按“C”键）。

第二行首的闪烁光标，表示等待用户输入过程体。所谓“过程体”就是我们用来绘图或解决问题而按照一定次序编排的命令序列，也是整个过程的核心。画边长为 40 步正方形的过程体就是如下的一组命令：

```
REPEAT 4 [FD 40 RT 90]
HT
```

在输入过程体时，每输入一行按一次回车键，这时电脑并不立即执行这一命令，只是把该行命令存入内存中，光标移到下一行的开头，等待继续输入命令。

过程体输入完毕，另起一行键入过程结束命令“END”。“END”在英文中为“结束”的意思，在此表示过程体的编辑结束。

最后，按“Ctrl－C”键，将清屏，返回混合屏状态。至此过程“ZFX”定义完毕。

介绍到这里，读者可能已经知道如何定义过程了，也知道一个过程应该包括几个部分。下面总结一下过程的格式。

定义过程实际上就是编制新命令，所谓“编制”，就是根据实际问题的要求，按照一定的格式和规则，使用 LOGO 命令，按一定的顺序编排成能够完成某种操作的一组命令，并给这组命令起一个能反映操作或状态的名称。这就好像我们日常生活中用“炒菜”来表示点火、放油、放菜、加佐料等一系列操

作一样。

过程的格式：

```
TO  过程名
  过程体
END
```

从过程的格式来看，一个过程分为三部分，即过程的首部(TO 过程名)、过程体、过程结束语(END)。

现在让我们执行刚才定义的绘制正方形的过程“ZFX”。键入：

? ZFX ↙

屏幕上立即绘制出一个正方形。清屏后，若再执行一次

? ZFX ↙

仍能绘制出相同的正方形。可见，过程已存在内存中，可以反复使用。现在，过程“ZFX ”就是一个绘制边长为 40 步正方形的命令。

在此提请大家注意的是：定义过程时，必须严格遵守过程定义格式，不能随意更改和简化。如果在定义过程前不键入 TO 命令，就不能进入过程定义状态，当然也就不能定义过程；在 TO 与过程名之间必须留有空格；LOGO 的原始命令不能作为过程名等等，否则电脑将作为错误命令处理。

四、定义过程示例

例 4—1　定义一个绘制小旗的过程，取名为“XQ”。

键入：

```
TO XQ
FD 35
```

```
REPEAT 2[FD 30 RT 90
FD 40 RT 90]
BK 35
HT
END
```

Ctrl－C　　（确认定义，退出编辑状态进入命令状态）

执行

```
? XQ ↙
```

将立即画出一面小旗。

例 4－2　定义一个画立体五角星（如图 4－2）的过程，取名为“WX”。

```
TO WX
HT RT 18
REPEAT 5[FD 60 RT 144 FD 6!
0 RT 54 FD 110 BK 110 LT 54!
+72]
END
```

例 4－3　定义绘制一只狐狸（如图 4－3）的过程，取名为“HULI”。这是一个比较复杂的过程，程序如下：

```
TO HULI
  HT
  LT 90 FD 80 RT 160 FD 73 RT 85 FD 28
  BK 28 LT 85 BK 52 LT 80 FD 25 LT 93
  FD 3 RT 108 BK 27 FD 27 LT 108 FD 7
```

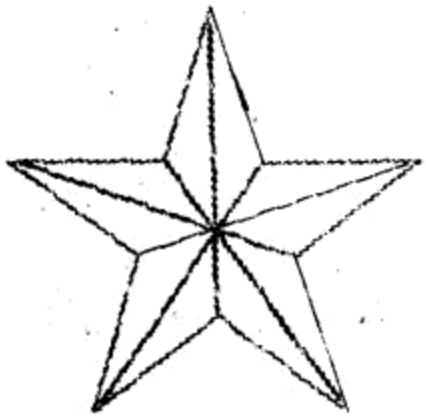


图 4—2　立体五角星

```
    RT 93 FD 40 RT 166 FD 55 LT 153
    FD 36 LT 105 FD 16 BK 16
    RT 105 FD 19 RT 165 FD 40 RT 95 FD 10
    PU RT 30 FD 12 PD
    FD 5 RT 140 PU FD 20 RT 130 PD FD 5
END
```

注——为压缩篇幅，每个语句行的屏幕换行时不再单独标出，读者可按语句行依次输入，系统会自动控制换行。

执行

```
HULI
```

将画出一只可爱的小狐狸。

到此为止，LOGO 命令集中已经扩充了四条命令，它们分别是“ZFX、XQ、WX、HULI”。如果你在命令状态下键入 ZFX 就能画出一个正方形、键入 XQ 就能画出小旗、键入 WX 就能画出五星、键入 HULI 就能画出一只狐狸。

图 4－3　狐狸

第二节　过程的嵌套

一、过程嵌套的概念

我们知道过程一经定义，就成为 LOGO 语言命令集中的一个新成员(新命令)，它可以像 LOGO 语言的原始命令那样被使用，使用时只要输入过程名即可。这种执行的方式，称为“过程调用”。

既然经过定义的过程可以当原始命令使用，那么它是否也能放置于另一个过程体中呢？回答是肯定的。

当一个被定义的过程名编排在另一个过程的过程体中时，就称为“过程的嵌套”。

二、过程嵌套示例

下面通过几个实例说明过程嵌套的使用方法。

1. 画一个“田”字

例 4－5　用过程调用的方法，定义一个画“田”字的过程。

我们已经在第一节定义了画正方形的过程 ZFX。现在就调用该过程画“田”字，取过程名为“TIAN”。定义方法如下：

```
TO TIAN
  REPEAT 4 [ZFX RT 90]
END
```

输入完过程后退出编辑状态，在命令状态下键入 TIAN，按回车键后，屏幕上将显示如图 4－4 所示的“田”字。在这个过程中出现了前面的过程名“ZFX”，我们称过程“TIAN ”嵌套了过程“ZFX”。

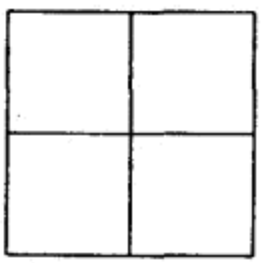


图 4－4 “田”字

细心的读者可能已经注意到：在过程“TIAN”中，我们并没有使用藏龟的命令，而海龟却不见了，知道这是什么原因吗？

通过嵌套画“田”字的过程，可以编写出绘制图 4－5 所示“双田”字图形的过程。见过程“TIAN2”。

```
TO TIAN2
  REPEAT 2 [TIAN RT 45]
END
```

通过此例可以看到，一经定义的过程即可被用来定义新的过程，使用新的过程又可定义新的过程。这样反复下去，就能像滚雪球似地不断扩大命令集合，不断增强 LOGO 的功能。

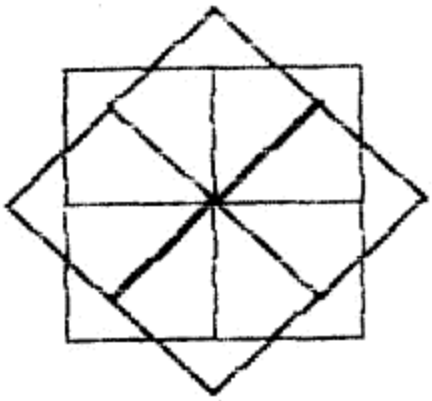


图 4—5 “双田”字

为了让读者能够熟练地掌握过程的嵌套，充分理解过程嵌套的作用。让我们继续看例题：

2. 画一把扇子

例 4—6 画扇子。

先编写一个过程“S”，画出两个一大一小的等边三角形套在一起。然后再编一个过程“SS”嵌套“S”过程，重复九次，每次转 10 度，就画出一把如图 4—6 所示的扇子。

过程如下：

```
TO S
  REPEAT 3 [FD 60 RT 120]
  REPEAT 3 [FD 90 RT 120]
END
TO SS
  HT
  LT 70
  REPEAT 9 [S RT 10]
END
```

注意——每个过程定义完毕，必须按“Ctrl－C”键确认后，才能编辑下一个过程。不能在一个过程的定义中输入多个过程的内容。

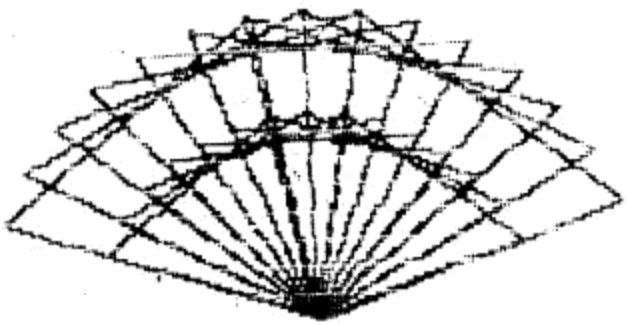


图 4－6　扇子

3. 画一朵盛开的花

例 4－7　画花

过程步骤：先编写画菱形的过程“LX”；再编写一个嵌套“LX”画出三个菱形的过程“SL”；最后再编写画花的过程“HUA”，让三菱形转出一朵花。

过程清单如下：

```
TO LX
  HT
  REPEAT 2 [FD 40 RT 60 FD 40 RT 120]
END

TO SL
  LT 30
  REPEAT 3 [LX RT 120]
```

```
END

TO HUA
  PU FD 50 PD
  REPEAT 8 [SL RT 45]
  LT 120 BK 120 RT 20 LX
  LT 100 LX RT 80 BK 50
END
```

键入“HUA”，回车后就可以画出一朵如图 4－7 所示的美丽花朵。

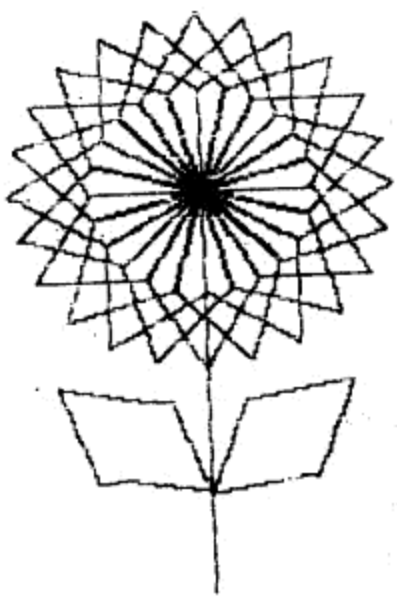


图 4－7　花

在过程“HUA”中，由画花过程“HUA”嵌套了画三菱形过程“SL”，而“SL”过程又嵌套了画菱形过程“LX”。除此之外，“HUA”过程还嵌套了“LX”过程，拼成一个完整的画花过程。

三、结构化程序的概念

上述用过程嵌套方法编程的思路是：首先把一个复杂的图形，分解成一个个简单、易画的基本图形；然后针对每个基本图形独立地编写、调试程序；最后根据整体图形的要求把各个小过程像搭积木那样拼装起来，组成一个完整的过程（也叫程序）。这种编程方法就称为“结构化的程序设计”。

结构化程序大大增强了 LOGO 语言的功能，使程序具有良好的结构性和较高的可靠性。

结构化程序的突出特点是：易读、易理解、易调试、易扩充。

四、结构化程序示例

下面我们再用过程的嵌套方法编写两个结构化程序实例，供读者欣赏。

1. 画蜘蛛

(1)定义蜘蛛的左脚：

```
TO LJ
  FD 40 LT 90
  FD 20 BK 20
  RT 90 BK 40
END
```

(2)定义蜘蛛的右脚：

```
TO RJ
  FD 40 RT 90
  FD 20 BK 20
  LT 90 BK 40
```

```
END
```

(3)定义过程“ZHIZHU”,调用已定义的过程“LJ”和“RJ”,完整的画出蜘蛛。

```
TO ZHIZHU
  DRAW HT
  RT 90
  REPEAT 4 [RJ LT 20]
  FD 30 BK 30
  LT 90
  REPEAT 4 [LJ RT 20]
END
```

键入“ZHIZHU”并回车,一个形态逼真的小蜘蛛将显示在屏幕上(见图 4－8)。

图 4－8　蜘蛛

2. **花窗图形**

(1)定义过程“A”,画出如图 4－9 所示的花窗基本结构图。

```
TO A
  FD 40 RT 90 FD 40 RT 90 FD 20 RT 90
```

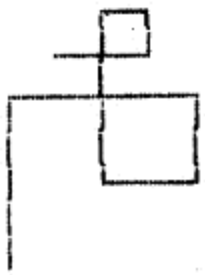


图 4－9　花窗基本结构图

```
  FD 20 RT 90 FD 40 RT 90 FD 10 RT 90
  FD 10 RT 90 FD 20
END
```

(2)调用过程“A”设计出各种图案

定义一个新过程，嵌套“A”过程，通过角度等变化，设计出各种花窗图案。

先看过程“B”，绘出的花窗图案如图 4－10 所示。

```
TO B
  HT
  REPEAT 8 [A LT 45 FD 70]
END
```

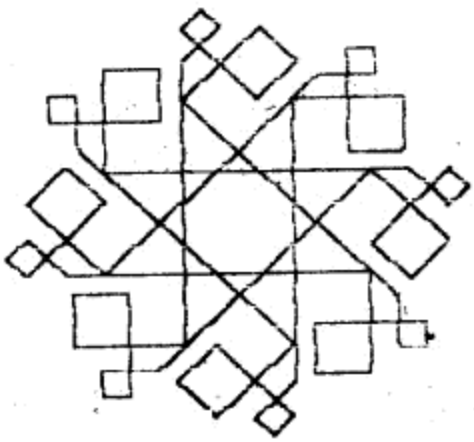


图 4－10　花窗图案(一)

过程“C”绘出的花窗图案如图 4－11 所示。

```
TO C
  HT
  REPEAT 4 [A A LT 90]
END
```

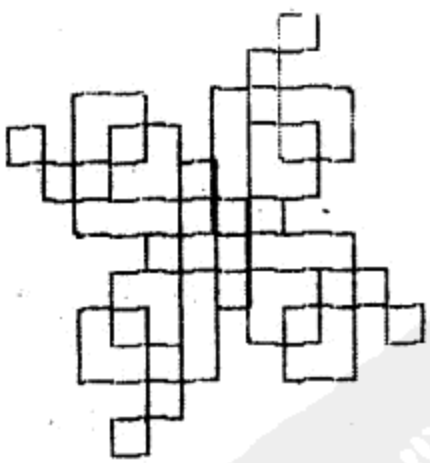


图 4－11　花窗图案(二)

第三节　出错信息

当输入命令、定义过程等操作中出现错误时，LOGO 系统均显示相应的信息给用户以提示。错误一般分为三类：语法错误、命令错误、过程错误。

一、语法错误

编写过程或程序时，必须严格遵守 LOGO 语言的各项规则（语法）。因为海龟作图是按照人们所规定的路线进行的，所以在输入过程时，不能随意向海龟发命令。如果你打入的命令违反了 LOGO 语言的语法规则，海龟就不能执行，它会向你发出“抱怨”，然后“罢工（停机）”。其实“抱怨”和“罢工”是一种积极的行动，它在“抱怨”中会告诉你发生了什么性质的错误、错误在什么地方。它停机的目的是为了让你改正错误。

例如，LOGO 语言的命令和参数之间应该间隔一个空格，这就是一条语法规则。当你把命令：

? FD 50

错误地打成：

? FD50

时，屏幕上会显示如下的错误信息：

PROCEDURE NOT FOUND IN LINE：

FD50

意思是说：“没有找到一个名字叫 FD50 的过程”。

因少打了空格，系统把“FD50”误认为是过程的名字。

二、命令错误

如果输入命令时敲错了字符键，或者输入命令参数的个数、种类不符合过程的定义要求，系统将报告出错信息，指明错误的性质。

请看下面三个示例：

(1)当把命令“HOME”错误地打成“HONE”时，屏幕上会显示如下的错误信息：

PROCEDURE NOT FOUND IN LINE:

HONE

意思是说：“没有一个命令或过程的名字叫 HONE”。检查刚输入的命令就会发现你错把字母“M”打成“N”。

(2)当把命令“PR 6+1”错误地打成“PR 6+”时，屏幕上会显示如下的错误信息：

FUNCTION NEED MORE INPUT IN

LINE: PR "6+

意思是说：“加号(+)后需要跟有参数”。

(3)当定义过程“TO CS”时，屏幕上会显示如下的错误信息：

FUNCTION NOT LIKE IT AS INPUT

ERROR IN LINE: TO CS

意思是说：“你定义的过程 CS 已经是 LOGO 的原始(固有)命令”，或者“与已定义的过程名字或命令重名”。

三、过程错误

在定义过程中，如果过程体中输入了错误的命令，系统将把它当做过程名处理，在定义时并不显示错误信息。仅在执行

该过程时，才报告存在的错误。如欲定义的过程是：

```
TO FANG
  REPEAT 4 [FD 50 RT 90]
END
```

在定义时，把过程中的命令“FD 50”误输入“FT 50”，当执行“FANG”时，显示：

```
PROCEDURE NOT FOUND IN LINE:
FT "50 RT "90
```

意思是“找不到 FT "50 RT "90 过程”。

当在执行过程“FANG”时，把过程名误打成“FAMG”时，将显示也将显示：

```
PROCEDURE NOT FOUND IN LINE:
FAMG
```

“找不到 FAMG 过程”。

错误信息详见附录(一)

第四节　过程的编辑

在编写一个比较复杂图形的绘图过程时，常常需要经过多次修改、加工才能完成。这就需要进行过程的编辑(修改)。

编辑过程必须在过程的编辑状态下才能进行。进入编辑状态的方法是：键入：

```
? TO 过程名
```

(这里的过程名就是要编辑修改的过程名)回车后屏幕上会显示出该过程的全部程序行。如果过程很长，一屏显示不下时，可以用“翻页键”找到要修改的程序页。

在编辑工作中，常用的编辑键及其作用如下：

一、光标移动键

1. →**光标左移动键**　该键能控制光标向左移动，而不改变光标所经过的任何字符。如果光标在程序行的最左端时，按该键会控制光标上移一行；当光标移到首行的第一个字符位置时再按→键，光标将不动。

2. ←**光标右移动键**　该键能控制光标向右移动，而不改变任何字符。如果光标在程序行的最右端时，继续按这个键会使光标下移一行；当光标移到最后一行尾时再按←键，光标将不动。

3. ↑**光标上移动键**　该键能控制光标向上移动一行，而不改变任何字符。如果光标在首行时按↑键，光标不动，并报警。

4. ↓**光标下移动键**　该键能控制光标下移一行，而不改变任何字符。如果光标在末行时按↓键，光标不动，并报警。

5. **光标快速左移键**“Ctrl—A”　这个组合键能控制光标快速移到本行开头。

例：TO…

……

FD 50 R ■ 90

↑

光标位置

按“Ctrl—A”后

TO…

……

■ 50 RT 90

↑

光标位置

6. **光标快速右移键**“Ctrl—E” 这个组合键能控制光标快速移到本行末尾。上例中按“Ctrl—E”后将出现

TO……

……

FD 50 RT 90 ■

↑

光标

二、删除字符键

1. **前删除键**“Esc” 删除光标左侧紧邻的一个字符，同时光标右边的字符全部左移一格。

例：

FD 100■

按一次“Esc”键后将出现：

FD 10■

2. **后删除键**“Ctrl—D” 删除光标位置的一个字符，同时光标右边的字符全部左移一格。

例：

FD 100

↑

光标

按一次“Ctrl—D”键后

FD 00

↑

光标

再键入 3，则变为：

FD 300

↑

光标

因此若需要在一行内插入字符，可先用光标移动键将光标移至插入处，然后键入所插入的字符即可。

3. **快速后删除键**“Ctrl—K” 删除光标右边的全部字符。

例：

FD 1000

↑

光标

按一次“CTRL—K”键后，

FD 1 ■

↑

光标

三、分行和并行

分行是把一行语句分成两行；并行是把两行语句合并成一行。

1. **程序分行键**↙(Enter) 在编辑状态下换行键可以作为分行键使用，按该键后，将以光标所在字符为首构成新的语句行。

例：

FD 50 RT 90 FD 40

↑

光标

按一次↙键后将出现：

FD 50

RT 90 FD 40

2. **合并程序行**　并行是这样完成的：先把光标移至上一行的末尾(可用 Ctrl—E 键)，再按 Ctrl—D 键(相当于删除换行键)即可。

例：

FD 50 ■

RT 90 FD 40

按一次 Ctrl—D 键后将出现：

FD 50 RT 90 FD 40

四、屏幕翻滚

以下键用于实现屏幕页面滚动：

Ctrl—B 键控制光标上移一屏，显示上面一页的内容。如果当前页就是第一页则光标移至屏首。

Ctrl—F 键使光标下移一屏，显示下面一页的内容。如果当前页就是最后的一页则光标移至屏尾。

Ctrl—L 将光标当前位置的内容卷至屏首。

五、退出编辑

退出编辑状态一般分为两种情况：

1. **保存编辑结果退出编辑状态**　对一个过程，当经过编辑修改，认为满意后，按“Ctrl—C”键(相当于过程定义结束)，即可将修改好的过程存入内存(也称工作空间)，如果内存中有同名的过程，则把它清除，存入新修改后的过程，并且退出编辑状态返回命令状态。

2. **不保存编辑结果退出编辑**　当进入编辑状态后，发觉

待编辑的过程不是欲编辑的过程时，按“Ctrl—G”键可退出编辑状态，刚键入的任何编辑命令都会作废，所作的一切修改均无效。如果内存中原来有同名的过程，则仍保留原过程。

Ctrl—C 键和 Ctrl—G 键的使用有十分重要的意义，屏幕编辑的结果是否保留，工作空间原有同名过程是否放弃都取决于这两个控制键的使用。

六、过程编辑小结

为了让读者更清楚地了解过程编辑修改的方法步骤，特总结如下：

(1)过程的编辑修改只能在编辑状态下、过程命令清单显示在屏幕上时进行。若当时是处于命令状态，要进入编辑状态后才能修改，应先打入命令：

TO 过程名(过程名是要修改的过程)

(2)使用光标移动键(需要时还可以用翻页键)把光标移到要修改的位置。

(3)删除字符时，要先把光标移至待删除字符的右侧，按 Esc 键，或者把光标移至待删除字符位置，按 Ctrl—D 键；插入字符时，要先把光标移至待插入字符的右侧，再输入待插入的字符；要把某些字符改为另外一些字符时，需要联合使用删除和插入两种功能。

(4)编辑工作完成后，要根据对编辑结果的取舍决定使用“Ctrl—C”或“Ctrl—G ”复合键。若欲保留编辑的结果则按“Ctrl—C”键；若放弃编辑结果就按“Ctrl—G”键；要想既保留当时编辑的结果又不破坏内存中原有的同名过程，就必须给编辑的过程另外起一个名字。

对初学者来说，不要求一下子全部记住这么多编辑功能

键的用法。可以先熟悉最基本的六个键“←、→、↑、↓、Esc、Ctrl－C”(这六个键足以完成各种修改),其它编辑键将随着学习的深入而逐渐熟悉。

混淆空格键和光标右移键是初学者常犯的错误,要通过实例弄明白它们的区别。虽然空格键也能控制光标右移,但在光标右移的同时,其所在位置的字符以及后面的所有字符都将右移,而光标的原来位置将插入一个空格。

另外,在编辑状态下,要把光标移到下一行时,应该用←键或用↓键,不要用↙键。

第五章　带参数的过程

我们知道，过程一经定义，就能像原始命令那样方便地使用。但是，以上定义的过程，使用范围太狭窄，如过程“ZFX ”只能绘制边长为 40 步的正方形，若绘制其它边长的正方形就要再重新定义。因此，LOGO 又提供了带参数的过程方式。带参数的过程是通用的绘图过程，根据参数的取值不同，一个带参数的过程可以绘制出无数个任意尺寸的相似图形，由此为过程赋予了广义的绘图功能。

本章将结合带参数过程的讨论，介绍 LOGO 坐标系的概念，并讨论下列指令的格式和使用方法。

海龟横向位移指令	SETX
海龟纵向位移指令	SETY
海龟横纵向位移指令	SETXY
海龟定向指令	SETH
海龟 X 坐标测试指令	XCOR
海龟 Y 坐标测试指令	YCOR
海龟指向测试指令	HEADING
画笔选择指令	PICKPEN
设置画笔颜色指令	PC
设置背景颜色指令	BG
设置颜色亮度指令	BRIGHT

第一节　参数及带参数的过程

一、参数的概念

参数在数学公式中很常见，如正方形的周长公式：

$$L=4a$$

对任何正方形都成立。边长“a”取一个确定的值，周长“L”就有一个确定的值。“a”发生变化，周长也随着发生变化，在这里“a”所表明的性质是正方形的边长，所以称它是周长的参数。显然，这里的参数是一个变量。所以在本书的讨论中，参数、变量是相同的概念，两种称呼共用。

前几章所编写的过程，都是在命令后面带有一定数值常数的语句，如“FD 40”、“RT 90”等。用带常数的命令定义过程，只能画出固定图形，也就是说所画图形的大小、形状是不变的。显然，要想使“ZFX”画出不同大小的正方形，就必须使前进命令“FD”后面的数值常数取不同的数值，即“FD”后面要取变量。

二、带参数的过程

过程体中，命令后面的常数由参数表示的过程称为“带参数的过程”。这种过程名的表达形式如下：

过程名　参数

过程名后面的参数，说明了过程体的命令中将使用该参数。同不带参数的过程名相比，多出了参数一项。由于现在的参数没有确定的数值，仅起到在过程名中占位子的作用，故通常称为“形式参数”；相应地当参数取确定数值后，就称为“实际参

数”。

带参数的过程也可以像 LOGO 原始命令或不带参数的过程那样直接调用，电脑将以

过程名　实际参数

的形式接受输入的命令，并执行这个命令。

三、定义带一个参数的过程

下面讨论定义带参数过程的方法

带一个参数的过程格式如下：

```
TO　过程名　参数名
  过程体
END
```

定义带参数过程的一般步骤是：

(1)键入“TO　过程名”进入编辑状态；

(2)按一下键使光标回到过程名的后面闪烁；

(3)按一次空格键后输入参数名，回车后参数就设定好了(这种定义参数的操作顺序比较特殊，特提请读者注意)。参数名由字母开头后面带若干字母或数字组成，在过程中可以把这些参数名当作变量来引用。

(4)输入过程体后退出编辑状态。

注意：在过程体中引用变量时，变量名的前面必须带有冒号。

下面以定义带参数的正方形绘制过程为例，介绍具体定义方法。

例 5－1　编一个大小可变化的正方形过程。

取过程名为“FX”，参数名为“BC”，带参数的过程如下：

```
TO FX BC
```

```
    REPEAT 4 [FD :BC RT 90]
    HT
END
```

定义操作过程：

(1) 键入“TO FX”，回车后，进入编辑状态；

(2) 按一次键使光标回到“FX”的后面闪烁；

(3) 键入空格和参数名“BC”，回车后，参数 BC 就设定好了；

(4) 输入过程体，按“Ctrl—C”退出。

至此带边长参数的正方形过程定义完毕。

现在，若要画一个边长为 20 步的正方形，只须键入：

```
? FX 20
```

回车后，一个边长为 20 步长的正方形就显示在屏幕上。

如果要画一个边长为 70 的正方形，只要打入：

```
? FX 70
```

回车后，就画出边长为 70 的正方形。

这样利用带参数的过程“FX”，可以方便地画出一组大小不同的正方形，如图 5—1 所示的七个正方形嵌套的图形，可定义如下的过程完成。

```
TO FX1
FX 10
FX 20
FX 30
FX 40
FX 50
FX 60
FX 70
```

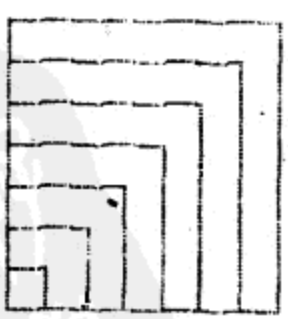


图 5—1　七个正方形嵌套图

END

执行“FX1”命令，即可画出图 5－1 的图形。

过程“FX1”有点繁琐，可以将它改写成“FX2”的形式。

```
TO FX2
MAKE "BC 10
REPEAT 7 [FX :BC MAKE "BC :BC+10]
END
```

执行“FX2”命令，仍显示图 5－1 的图形。两个过程虽然功能相同，但形式差别很大，显然，“FX2”的结构较紧凑。

过程“FX2”中的“MAKE”是新面孔，它是 LOGO 语言中固有的赋值命令。具体的使用方法及作用将在后面章节介绍。

参数还可以用来决定图形的形状、种类。按照前面曾讲过的正 N 边形，重复次数是 N，转角是 360/N 度的规律，只要把边数作为变量，就可以编写一个画任意正多边形的通用过程。

例 5－2　编写画任意正多边形的通用过程。过程名为“ZDBX”。

过程如下：

```
TO ZDBX N
  REPEAT :N [FD 40 RT 360/:N]
END
```

这里，命令“RT”后面的参数值是一个算式，执行时电脑先算出转角值，再执行右转向命令。一般来说，命令所需要的参数值都可以用变量或算术表达式赋给。

在带参数过程“ZDBX N”中，令参数 N＝3 就画出正三角形，参数等于 4、5、6、7、8 就分别画出正四、五、六、七、八边形（见图 5－2）。

上述“ZDBX”过程定义好后，可编写一个变化过程来实

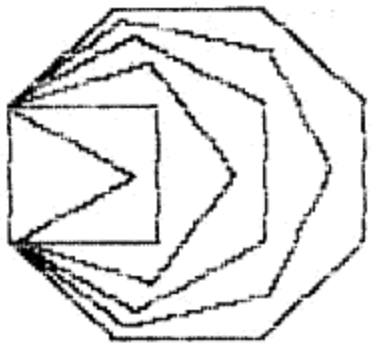


图 5—2　正多边形的嵌套

现绘制边长为 40、边数不同的一组正多边形。过程如下：

```
TO BH2
  ZDBX 3 ZDBX 4 ZDBX 5 ZDBX 6 ZDBX 7 ZDBX 8
END
```

四、带参数过程的一般格式

参数的功能是多种多样的，凡是可以用量(各种类型的量，常量、变量等)描述的属性都可通过参数来表述。一个过程中的参数也可以超过一个，带参数过程的一般格式为：

```
TO  过程名  参数 1  参数 2  参数 3  ...  参数 N
    过程体
    END
```

这种格式和不带参数的过程基本相同。它包括三部分：

1. **标题行**　指出过程的名字和各参数的名字。过程名与参数名之间要用空格隔开；各参数名之间也要用空格隔开。

2. **过程体**　由一个或多个 LOGO 命令行组成，或者说是 LOGO 命令的序列。构成过程体的 LOGO 命令，可以是固有

命令，也可以是过程。

在定义过程时既可以引用已定义的过程，也可以调用还没定义的过程。这样就摆脱了程序编写时对过程编写顺序的严格限制，可以在编写完主过程后，再编写副过程。

3. **结束行** 结束行“END”应单独占一行。定义过程时，如果没有输入“END”命令就按复合键“Ctrl—C”，LOGO 系统会自动添加结束行“END”。

五、带参数过程示例

例 5－3 画平行四边形。

分析：平行四边形，由相邻的两边边长和它们之间的夹角确定。因此，编写画任意平行四边形的过程，就要有三个变量分别表示两个边长和一个转角。取过程名为“PS”、两边长的参数名分别为“B1”、“B2”，角参数名为“J”，编写过程如下：

```
TO PS B1 B2 J
  HT
  REPEAT 2 [FD :B1 RT :J FD :B2 RT 180－:J]
END
```

执行过程“PS”时，必须给出三个参数值。比如输入：

```
RT 30 PS 40 70 60
```

即可画出如图 5－3 所示的平行四边形。

六、带参数过程的嵌套

带参数过程也可以嵌套使用（调用时注意给参数赋值）。

例如，嵌套“PS”过程画一个如图 5－4 的长方体。过程如下：

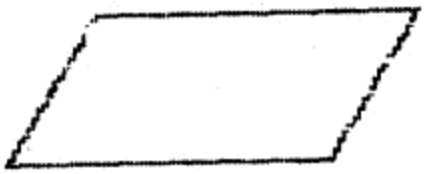


图 5－3　平行四边形

```
TO CFT
  LT 90 PU BK 50 PD
  PS 120 50 140
  LT 90
  PS 50 120 90
  LT 130
  PS 50 50 130
END
```

其中调用 PS 过程时后面给出了三个参数值。

执行“CFT”命令，将画出图 5－4 所示的长方体。

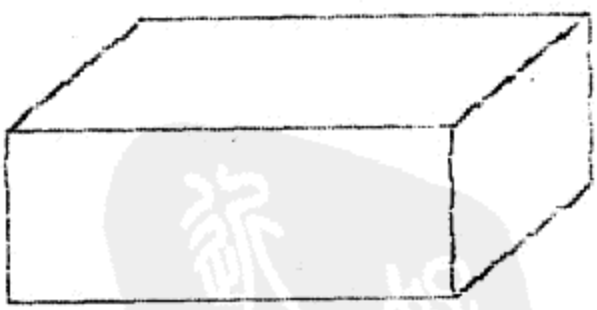


图 5－4　长方体

在长方体的基础上再添上几块长方形和一朵花就画出一个如图 5－5 所示的电视机。过程如下：

```
TO DSJ
  PU FD 60 LT 90 BK 50
  PD PS 130 40 140
  LT 90 PS 100 130 90
  LT 130 PS 40 100 130
  RT 40 BK 6 PU RT 90 FD 6 PD
  PS 50 15 90 PU FD 88 RT 90 PD
  PS 15 10 90 PU FD 21 PD
  PS 97 88 90 PU FD 49 RT 90 FD 44
  PD REPEAT 6 [PS 15 15 40 RT 60]
END
```

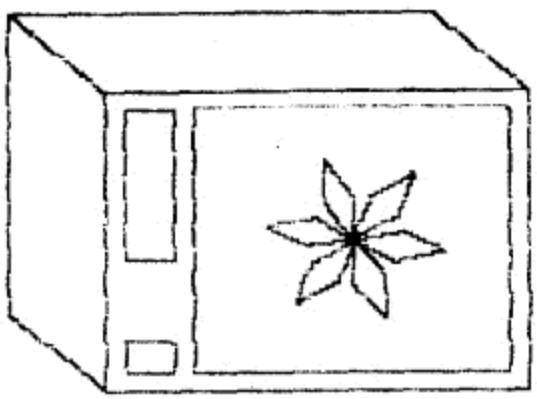


图 5－5　电视机

与不带参数的过程相比，带参数过程功能更强、使用更灵活。

带参数过程可以替代已经介绍过的各种绘图过程。

第二节　形式参数和实在参数

LOGO 中有两类参数：形式参数、实在参数。对这两种参

数各有什么要求？应具备什么样的条件呢？

1. **形式参数** 在带参数过程定义中，写在过程后面的参数都是形式参数。如上一节“PS”过程中的“B1、B2、J”都是形式参数。

对于形式参数的命名没有什么特殊要求，只要满足以下两个条件就可以：

(1) 参数名字是不带双引号的 LOGO“字”；

(2) 同一个过程的各个形式参数不能重名，但不同过程的形式参数可以同名。形式参数即使同其它过程名、甚至固有命令名重复也没有关系。

形式参数是过程定义时、过程体中所用变量个数的清单。它们的具体数值在被调用、执行时确定。

2. **实在参数** 在调用、执行一个带参数过程时所使用的常数叫做实在参数。例如，画边长为 40 的正七边形，可以键入命令：

```
ZDBX 7
```

这里的“7”就是实在参数。

注意——常数做实在参数时不要加冒号。如上例中千万不要写为“:7”。

实在参数也可以是变量或者算术表达式。实在参数取变量形式时，其前面的冒号不能缺少，而且必须是已经赋值的变量。

第三节 圆和圆弧

圆和圆弧也是基本图形。用圆可以组出许多美丽的图案；用圆弧可以拼成各种各样的曲线。

一、圆

我们知道，海龟只能画水平和垂直的直线，画出的斜线就已经是锯齿形的，若让它画出一个真正的圆就更困难了。也就是说，海龟不能画出真正的圆。

那么我们怎样来画圆呢？

前面已经介绍过用重复命令画圆的方法，其实就是用一个边数很多的正多形代替圆。例如，边长很短的正三百六十边形近似圆形。于是我们可以编写画圆的通用过程如下：

```
TO YUAN1 B
  REPEAT 360 [FD :B RT 1]
END
```

过程“YUAN1”为画圆的通用程序，使用它我们可以选择实在参数“0.1、0.2、0.5、0.8、……”等画一系列内相切的圆。

二、画圆公式的导出

人们画圆，通常习惯是确定半径画圆，很少用设定边长的方法画圆，所以过程“YUAN1”使用很不方便。因此，必须编写用半径控制画圆的过程。

因为 LOGO 以正 360 边形近似圆，那么该多边形的周长与圆的周长有以下关系：

$$360\times B=2\pi R$$

式中“B”为 360 边的边长、“R”为圆的半径，若圆周率“π”取近似值 3.14159，代入上式得：

$$B\approx 3.14159\times R/180\approx 0.0174\times R$$

因此过程“YUAN1”可以修改为：

```
TO YUAN2 R
```

```
    REPEAT 360 [FD 0.0174 * :R RT 1]
END
```

(过程体中的“*”是乘法运算符号、“/”是除法运算符号)。

使用过程“YUAN2”,可以画出任意半径的圆。

现在画一个半径为 30 的圆。

键入:

```
.ASPECT 1.2
YUAN2 30
```

海龟在屏幕上画出如图 5－6 所示的圆。这个圆的半径就是 30。让我们来验证一下,键入:

```
RT 90 FD 60
```

回车后,海龟先向右转 90 度,然后画出一条长为 60 步的直线(见图 5－7),这条直线恰好就是圆的直径,这就证实了圆的半径等于 30。

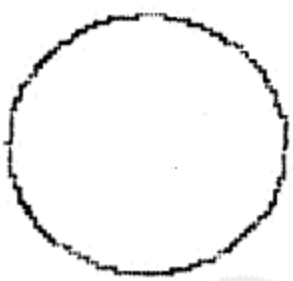


图 5－6 圆

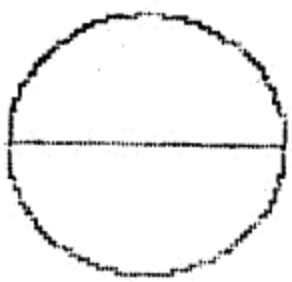


图 5－7 圆的直径

三、提高画圆速度

由画圆过程可见,画圆的速度非常缓慢。原因是海龟每走 1 步就要旋转 1 度,重复次数太多。显然,要想加快速度,必须减少重复次数,即让海龟连续走几步再转几度。如让海龟走 5

步后再转 5 度，过程如“YUAN3”。

```
TO YUAN3 R
  REPEAT 72 [FD 0.0174 * 5 * :R RT 5]
END
```

或者，让海龟走 10 步再转 10 度，过程如“YUAN4”。

```
TO YUAN4 R
  REPEAT 36 [FD 0.174 * :R RT 10]
END
```

用过程“YUAN4”所画的圆如图 5—8 所示。

画圆过程的修改原则仍是用正 N 边形代替圆，重复次数是“360/N”。“YUAN4”取重复 36 次画出的圆形稍粗糙一点，但由于使用电视机作显示器，分辨率不高，取正 36 边形代替圆已很近似了。

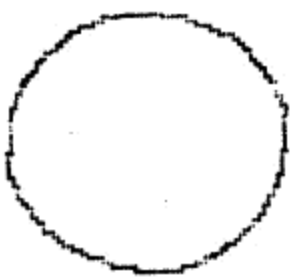


图 5—8　36 边形的近似圆

在定义画圆过程时，还要注意一点小技巧。仔细观察图 5—8 中的圆稍微有点歪，这是由于画圆的起点和终点状态不一致造成的，只要将过程“YUAN4”中的过程体改为：

```
REPEAT 36 [RT 5 FD 0.174 * :R RT 5]
```

就可纠正。于是我们得到最终的画圆过程“YUAN”。

```
TO YUAN R
```

```
  REPEAT 36 [RT 5 FD 0.174 * :R RT 5]
  HT
END
```

上面所讨论的画圆过程，都是按顺时针方向进行的，称为“右画圆”。只要将右画圆过程体中的“RT”改为“LT”即可按逆时针方向画圆，称为“左画圆”。读者不妨试一试。

四、圆弧

“弧”可以看作圆周的一部分，画圆使海龟旋转 360 度，画弧则只要让海龟转到所需的角度就可以了。因此，画圆弧过程需要半径(R)和重复次数(N)两个参数。定义圆弧的过程见“YH”。

```
TO YH N R
  REPEAT :N [RT 5 FD 0.174 * :R RT 5]
  HT
END
```

过程“YH”可绘出“右画弧”。

若把过程体中的“RT”改为“LT”，就得到“左画弧”的过程“ZH”。

```
TO ZH N R
  REPEAT :N [LT 5 FD 0.174 * :R LT 5]
  HT
END
```

执行上面过程，输入命令“YH 18 60”就画出半径为 60 向右画出的半圆弧(见图 5－9)；输入命令“YH 9 60 ”就画出了半径为 60 的 90 度右画弧(见图 5－10)；输入命令“ZH 15 70 ”就画出半径为 70 的 150 度左画弧(见图 5－11)；输入命令

“ZH 27 60 ”就画出半径为 60 的 270 度左画弧(见图 5－12)。

图 5－9　右画弧(一)

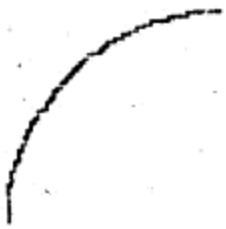


图 5－10　右画弧(二)

图 5－11　左画弧(一)

图 5－12　左画弧(二)

第四节　圆和圆弧过程的应用

圆和圆弧的过程是很有用的基本过程，为加深对圆、圆弧的认识，掌握圆、和圆弧的应用，本节列出几个应用实例供参考。

一、幼苗

例 5－4　画一棵如图 5－13 所示的幼苗。首先定义画弧过程“YH”，然后再定义幼苗过程“YM”。

```
TO YH N R
  REPEAT :N [RT 5 FD 0.174 * :R RT 5]
```

```
END

TO YM
  REPEAT 2 [YH 9 40 RT 90 YH 9 40]
  FD 40
  HT
END
```

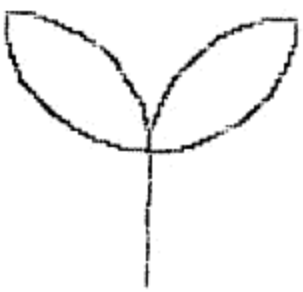


图 5—13　幼苗

执行过程“YM”就可以画出幼苗。不过这棵幼苗的大小是固定的。要想控制它的大小，可将过程“YM”改成过程“YM1”。

```
TO YM1 R
  REPEAT 2 [YH 9 :R RT 90 YH 9 :R]
  FD 40
  HT
END
```

二、五瓣花朵

例 5—5　画一朵如图 5—14 所示的花，过程名为“HUA”。

```
TO HUA
  REPEAT 5 [HB RT 72]
  HT
END

TO HB
  REPEAT 2 [Y 9 25 RT 90]
  RT 45 FD 10 LT 90 Y 36 2
  RT 90 BK 10 LT 45
END

TO Y N R
  REPEAT :N [RT 5 FD 0.174 * :R RT 5]
END
```

过程“HB”的功用是画一片花瓣。

在命令状态下输入“HUA”之后，一朵美丽的小花将展现在你的眼前。

图 5－14　五瓣花朵

三、太阳

例 5－6　画出如图 5－15 所示的太阳。过程如“TY”。

```
TO TY
  REPEAT 12 [YH 3 40 LT 90 PU FD 10 PD FD 15
  PU BK 25 PD RT 90]
END
```

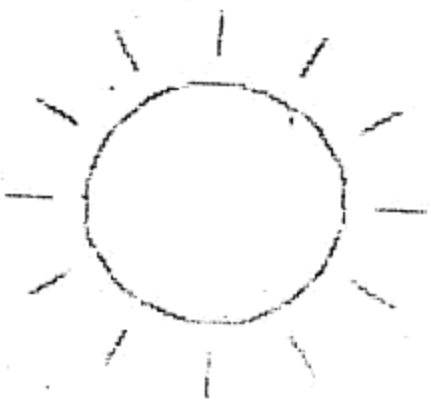


图 5－15　太阳

方括号中命令的功用是画一小段弧和一道光芒。其中“YH”是例 5－4 中的过程。

四、葵花

例 5－7　画出如图 5－16 所示的葵花。过程名为“KH”。

```
TO L :X :B :J
  REPEAT :X [FD :B RT 150 FD :B LT :J]
END

TO KH
  HT
```

```
  PU FD 80 PD
  REPEAT 18 [YH 18 10 LT 160]
  PU BK 105 PD RT 25 L 45 13 158
  LT 12 PU FD 40 PD L 12 25 180
END
```

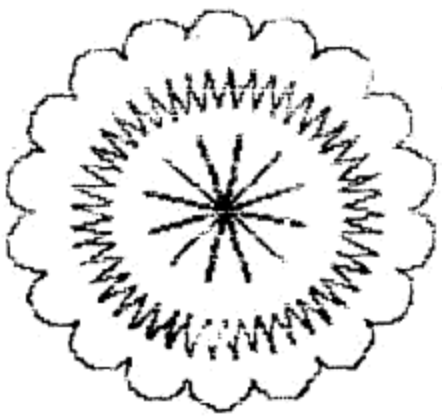


图 5—16　葵花

过程“KH”中，“L ”过程的功用是画锯齿形花纹和中心射线；过程“KH”先调用“YH”过程画出圆弧组成的轮廓线，然后两次调用“L”过程取不同的参数分别画出锯齿形花纹和中心射线。

五、小兔

例 5—8 画一只奔跑的小兔(如图 5—17 所示)，过程名为“TU”。

```
TO ZYH N R
  REPEAT :N [LT 5 FD 0.174 * :R LT 5]
END

TO YYH N R
  REPEAT :N [RT 5 FD 0.174 * :R RT 5]
```

```
END

TO TU
  PU LT 90 FD 60 RT 90 PD
  YYH 36 15 RT 90 PU FD 8 PD LT 90
  YYH 36 4 LT 90 PU FD 8 PD RT 90
  YYH 10 15 LT 90 YYH 9 20 RT 90
  YYH 18 20 RT 90 YYH 9 20 LT 90
  YYH 16 15 LT 15 YYH 6 45 RT 190
  ZYH 7 85 RT 40 ZYH 6 70 LT 168
  YYH 4 50 YYH 3 10 YYH 10 6 RT 10
  FD 10 LT 125 ZYH 6 46 YYH 6 42
  HT
END
```

图 5—17　奔跑的小兔

通过上述五例，大家一定体会到圆弧过程的妙用。圆弧过程中的两个参数“N 和 R”，可以用来控制弧的半径及弧长，最终改变曲线的形状。

在绘制比较复杂的图形时，准确地控制弧的半径、弧长参数来控制曲线的形状是件不容易的事，需要不断地实践和练

习，才能真正掌握。

第五节　多角星

一、奇数角星的画法

同画正多边形一样，画多角星的关键也是计算转角。多角星计算转角的公式为：

当角数“N”是奇数时，重复次数取 N，

转角＝180－180/N

例如，我们要画如图 5－18 所示的九角星，根据上面的公式可计算出海龟所转的角度为：

180－180/9＝160 度

图 5－18　九角星

过程名取“JJX”，定义过程如下：

```
TO JJX B
  REPEAT 9 [FD :B RT 160]
END
```

键入“JJX 70”就可以画出图 5－18 所示的九角星。

当欲画偶数角星时（如八角），必须将上面的过程改成：

```
TO BJX B
  REPEAT 8 [FD :B RT 180-180/8]
END
```

执行“JJX 70”只能画出图 5－19 所示的图形。

图 5－19　初级八角星

二、多角星的讨论

朋友们可以实践一下，能否一笔画出任意个角的多角星呢？实践发现，这确是十分困难的事情。

现在我们来讨论一笔画多角星的一般规律。

1. 多角星的一般公式

设角数为“n”、每一角的内角为“α”，它们之间存在如下关系式：

$$\alpha=(n-2k)180/n$$

其中“k”为正整数，$1<k<n/2$，且 k 与 n 没有公因数。

2. 公式的证明

取 n＝12、k＝5，将圆 n(12)等分(见图 5－20)，

从顶点 A0 开始，沿逆时针方向与 A0 以下的第 k 个点(A5)Ak 连一条直线，再从 Ak 开始与 Ak 以下的第 k 个点(A11)连一条直线……，这样依次连下去。每条连线对应的弧

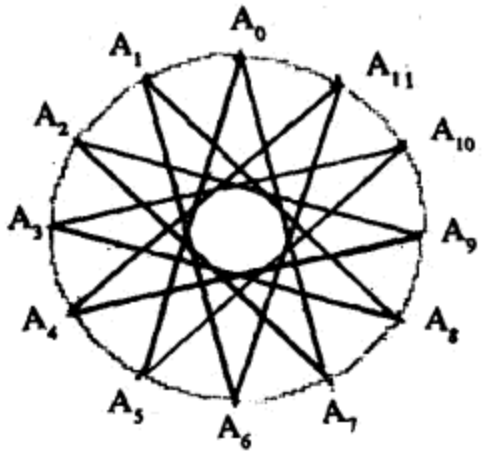


图 5－20

度为

$$2k\pi/n \qquad (2\times 5\pi/12)$$

（点从 A0 到 A5 这段弧长），这时，多角星的内角恰好为

$$(n-2k)180/n \qquad ((12-2\times 5)\times 180/12=30°)$$

进行 n(12)次，点跑过的弧度为 2kπ(2×5π)，即跑了 k(5)圈。因 k 是正整数，因此必然回到起始点 A0。这就证明了公式的存在。

那么，有没有连线不到 n 次，点就回到起始点的情况呢？我们再来证明这一点。

按上述方法连线，假定进行了 m 次(m＜n)，就已经回到起始点 A0(以后就会重复画，而不再经过剩下的顶点)，这样画出来的将是 m 角星，这时，点跑过的弧度为

$$2k\pi m/n$$

它必是 2π 的整数倍，即有

$$mk/n=1$$

$$k/n=1/m$$

而 m＜n，所以 k 与 n 必有公因数，与公式的条件相违背(如取 k＝ 4 时，将画出三角形，即 m＝3，k、n 存在公因数)。

反过来，如果 k 与 n 没有公因数，对于 m＜n，km/n 就不会是正整数，也就是说进行 m 次不会回到 A0 点，只有进行 n 次，才会回到 A0 点，反证公式亦成立。

如果沿顺时针方向依次连线，与上面的情形一样。

上述证明比较难懂，如果看不明白也不必花太多精力，只要会用这个公式就行了。

3. 多角星通用过程

依据多角星公式，当 n＝3、4 时多角星与正多边形相同；当 n＝6 时，不可能一笔画出多角星，这是因为满足 1＜k＜3 的唯一一个正整数 2 又与 6 有公因数，即没有一个 k 可以满足公式的要求；当 n 为其它任何正整数时，皆可按上述公式确定内角，画出正 n 角星。

根据上述规律我们可以编写任意多角星的过程“DJX”。注意：转角应为外角，它的计算公式为：

$$180-180(n-2k)/n=360k/n$$

过程如下：

```
TO DJX N B k
  HT
  REPEAT :N [FD :B RT :k * 360/:N]
END
```

过程中有三个变量，分别为：边长 B、角数 N、k 均为公式中的参数。这样画任意多角星的过程就定义好了，以后在画多角星时只要调用这个过程即可。

4. 画多角星

前面讨论的奇数角星，是多角星的一种特述情况。例如画边长为 70 海龟步的 11 角星。由

$$1<k<11/2$$

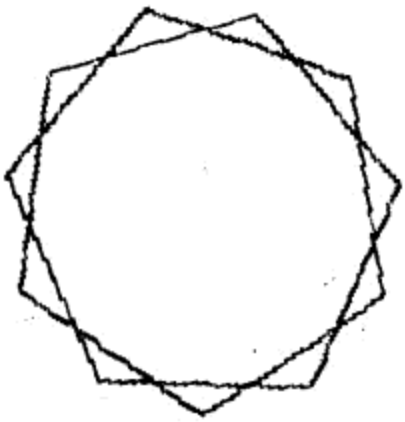
可算出与 11 没有公因正整数有

2,3,4,5

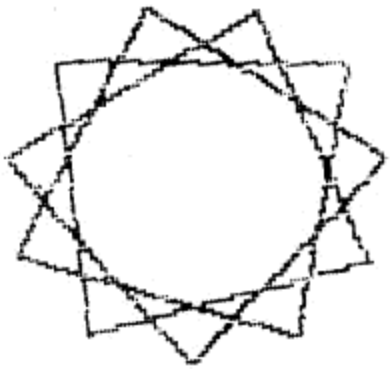
分别令 k =2、3、4、5,执行

DJX 11 70 k　　(k=2,3,4,5)

可画出如图 5—21 所示的 11 角星。

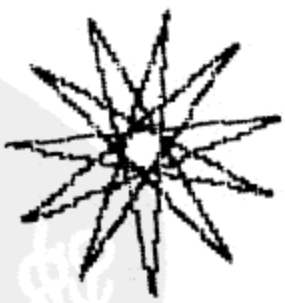


k=2 时



k=3 时

k=4 时



k=5 时

图 5—21　十一角星

三、多角星绘制实例

为了使读者能够熟练掌握任意角星的绘制方法，下面列出几个实例供参考。

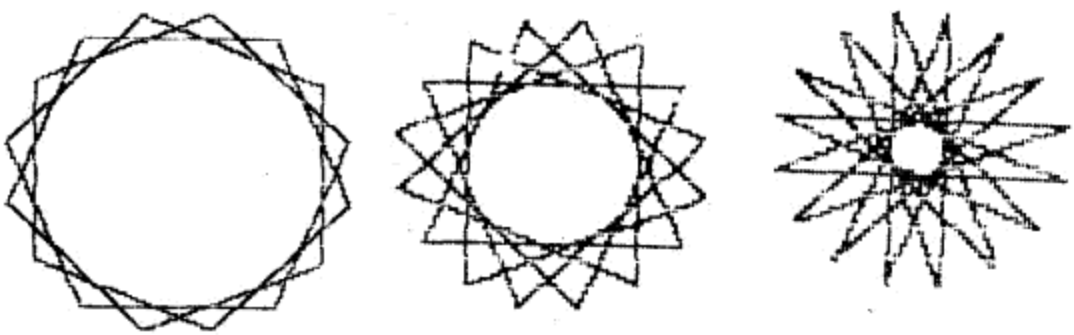
例 5－9 画边长为 70 的 16 角星。

分析：1＜k＜16/2，且与 16 没有公因数的正整数有 k＝3,5,7。

键入：

? DJX 16 70 k

（k 为 3,5,7）便可画出图 5－22 所示的三个图形。



k＝3 时　　k＝5 时　　k＝7 时

图 5－22　16 角星

例 5－10　以最小的内角画边长为 50 的 24 角星。

0＜k＜12，与 24 没有公因数的有：

k＝5,7,11。

由图 5－21 可知，当 k 取最大值时，多角星的内角最小，因此我们取 k＝11。键入：

? DJX 24 50 11

便可画出图 5－23 所示的图形。

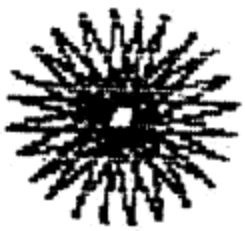


图 5－23　内角最小的 24 角星

四、多角星的应用

例 5－11　使用多角星画出图 5－24、5－25 所示的图形。

过程分别见“X1”、“X2”。

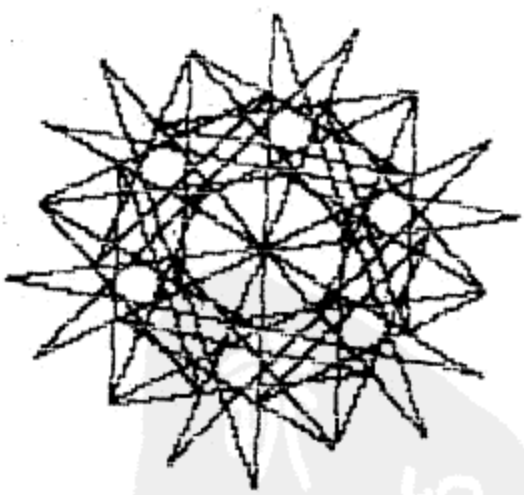


图 5－24　复杂花窗图案(一)

```
TO X1
  REPEAT 6 [DJX 9 80 4 RT 60]
END
```

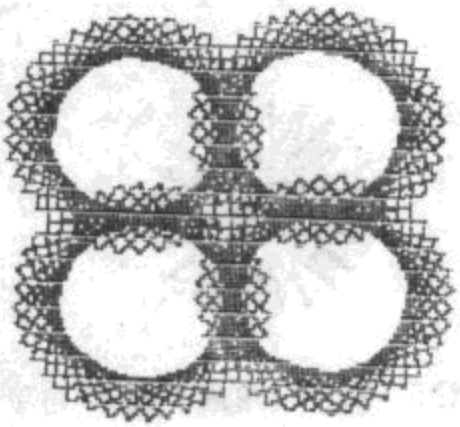


图 5－25　复杂花窗图案(二)

```
TO X2
  REPEAT 4 [DJX 40 80 11 RT 90]
END
```

执行“X1”可画出图 5－24 的图形；执行“X2”将画出图 5－25 所示的图形。

第六节　海龟的坐标与定向

在 LOGO 的三种屏幕显示方式中，图形屏和混合屏是供海龟作图的，是海龟活动的场所。它在屏幕内上、下、左、右各走的步数如图 5－26 所示。

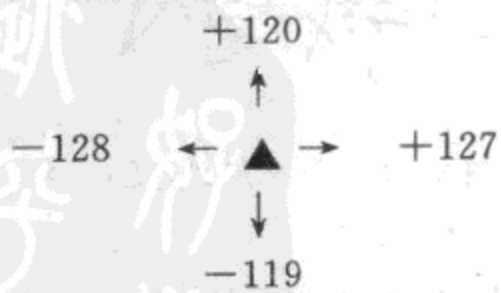


图5－26　海龟绘图空间坐标示意图

在这个空间里你可以充分发挥自己的想象力，施展自己的绘图才能，创造一幅幅美妙的图案。但是，往往因数学知识所限，对有些图形，一时不能算出海龟所转角度和前进的步数，编程中只能通过反复试探确定，致使编程效率降低。例如输入命令：

? RT 90 FD 35 LT 90 FD 20

将画出直角三角形的两条直角边(如图 5－27 所示)。

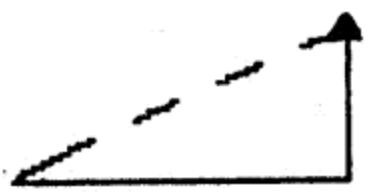


图 5－27　直角三角形

要画直角三角形的斜边，若没有三角函数的知识，可能要试探多次才能确定。

再如，当进行两幅图形的叠加时，因不知道绘完第一幅图形后海龟在屏幕上的状态(方位和指向)，也只有多次试探才能确定下一幅图形的绘制方式。

为提高绘图编程技能，必须掌握海龟的定位、定向指令。

一、直角坐标系

为使没有学过坐标的读者也能使用定位命令，我们先从直角坐标系谈起。

1. **数轴**　在一条直线上规定了原点“O”，正方向和单位长度，就构成了数轴。

2. **坐标** 在数轴上每个点的位置都能用一个实数来表示，这个实数叫做这个点在数轴上的坐标。

定义了数轴和坐标后，任何一个实数都可以在数轴上找出它对应的坐标。也就是说，数轴和坐标可以唯一确定直线上的一个点。

3. **坐标系** 我们知道，一个数轴可以表示直线上的一个点。那么，平面上的一个点应如何表示呢？众所周知，电影院的座位是用“排、号”两个数来表示它所在位置的，观众按照这两个数就可以找到自己的座位。实际上，座位就是平面上的一个点，“排”代表座位的纵向坐标、“号”代表座位的横向坐标，两个坐标的交点就是座号。数学中的平面坐标系就是从“排、号”的概念中抽象出来的。在平面内画两条互相垂直而且相交的两个数轴（如图 5－28 所示），其交点称为坐标原点“O”、水平方向的数轴称为“X”轴（或横轴）、垂直方向的数轴称为“Y”轴（或纵轴），两条数轴上取相同的单位长度，这样就构成了平面直角坐标系。

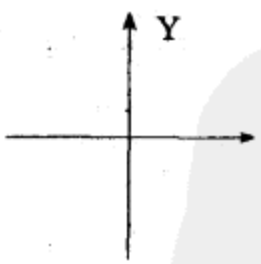


图5－28

平面上建立了直角坐标系后，平面内的任意一点，都将有一对有序实数与它对应。如对于点 p（见图 5－29），经过点 P 画 X 轴的垂线，垂足落在 X 轴上的刻度 3 上；再经过 P 点画

Y 轴的垂线，垂足落在 Y 轴上的刻度 2 上，这样，点 P 就有一对坐标为(3,2)和它对应。我们把 3 叫做点 P 的 X 坐标，2 叫做点 P 的 Y 坐标，和起来叫做点 P 在平面内的坐标，记作 P(3,2)，其中 X 坐标规定写在 Y 坐标的前面，中间用逗号隔开。这就是说，点 P 在平面内有唯一的一对有序数与其对应。

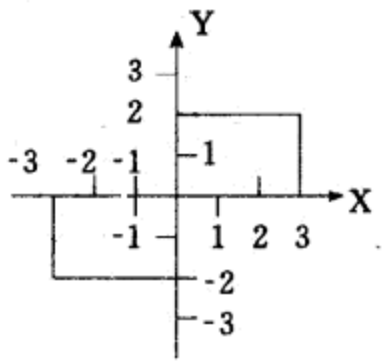


图5－29

反过来，对于任意一对有序数(实数)，在坐标平面内都有一个确定的点与它对应，这个点在平面内的坐标就是这一对有序实数。例如，对实数对(－3，－2)，我们可以经过 X 轴上坐标为－3 的点画 X 轴的垂线，经过 Y 轴上坐标为－2 的点画 Y 轴的垂线，这两条的交点 C 就是(－3，－2)对应的点。同样也可以找出其它有序实数对在平面坐标系中的位置。因此，坐标平面内所有的有序实数对之间是一一对应的。

二、LOGO 的绝对坐标

在以上讨论的绘图中，海龟的前进、后退步数，都是以当前海龟所处位置为参照点描述的；海龟转动的角度也是以海龟的当前方向为依据旋转的，这是相对坐标的概念。

1. **绝对坐标的引入**　使用相对坐标编写绘图过程，没有坐标计算，入门容易，编程简单，特别适合初学者学习。但当编

写多个图形叠加程序时，相对坐标将显得捉襟见肘。若使用绝对坐标则十分容易。

2. **绝对坐标系** LOGO 的绝对坐标系如图 5－30 所示。坐标原点选在屏幕中心(海龟的母位)；X 轴又称为横轴，从原点向右为正，向左为负；Y 轴又称纵轴，从原点向上为正，向下为负。坐标的绝对值，用海龟爬行的步数表示。如果海龟从中心开始，向右走 50 步再向上走 30 步，所停留点的坐标用(50，30)表示，它说明海龟横坐标等于 50、纵坐标等于 30。再如，由原点向左走 70 步，再向下走 50 步，停留点的坐标就是(－70，－50)。显然，LOGO 的绝对坐标系就是上面介绍的直角坐标系。

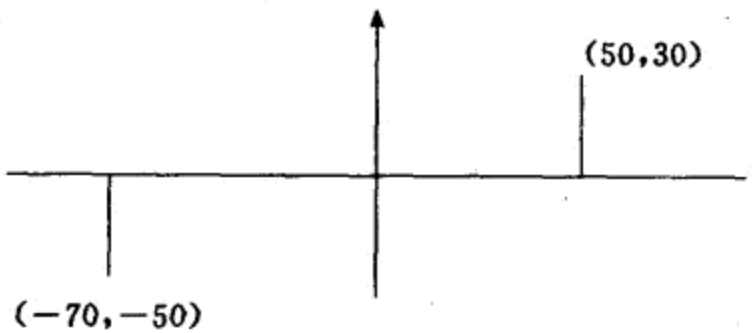


图5－30 绝对坐标系

在绝对坐标系中描述海龟在屏幕上的状态时，还需要给出海龟的尖头方向。方向的绝对坐标零度基线是正上方，顺时针为正，逆时针为负。尖头指向正上方的绝对方向角为零度；指向正右方时的绝对方向角为 90 度；指向正左方时的绝对方向角为 270 度或取补角称为－90 度等等。

三、海龟的坐标位移指令

使用以下几条指令可以把海龟移到指定的位置。

1. **海龟横向位移指令** SETX

格式:SETX　参数

类型:语句、命令

功用:控制海龟横向位移到指定位置。

注释:SETX 来源于"SET X coorolinate",意思是"设定海龟在 X 轴方向的坐标"。

参数是海龟横向移动终点的 X 坐标值。它可以取常数、变量或算术表达式的形式,取值范围为±10 的±18 次方。

因 X 轴以 256 为周期,当位移距离超过 256 时,虽然海龟要移规定的步数,但在屏幕上的显示位置将为除以 256 的余数(注——若是落笔状态时,将画出一条横贯屏幕的直线)。如输入

? SETX 256

回车后,画完一条横贯屏幕的直线后,海龟仍停留在原位。

说明:该命令控制海龟沿水平方向,移到 X 坐标为参数所指定的位置上。移动过程中,海龟的 Y 绝对坐标值不变,海龟尖头方向不变,海龟所经过的直线轨迹是否画出,取决于提笔、落笔、笔色的状态。

示例:

? SETX 50

回车后,海龟向右移 50 步,若当前为落笔状态,将画出一条长 50 的直线。

2. **海龟纵向位移指令** SETY

格式:SETY　参数

类型:语句、命令

功用:控制海龟纵向位移到指定位置。

注释:SETY 来源于"SETY coordinate",意思是"设定海龟在 Y 轴方向的坐标"。

参数是海龟纵向移动终点的 Y 坐标值。它可以取常数、变量或算术表达式的形式，取值范围为±10 的±18 次方。

因 Y 轴以 240 为周期，当位移距离超过 240 时，虽然海龟要移规定的步数，但在屏幕上的显示位置将为除以 240 的余数(注——若是落笔状态时，将画出一条纵贯屏幕的直线)。

说明：该命令指示海龟沿垂直方向移到 Y 坐标为参数所示的位置上。这里的参数是终点的 Y 坐标值，不是指海龟所走的步数。运动过程中，X 绝对坐标值不变，海龟尖头方向不变。海龟所经过的直线轨迹是否画出，视提、落笔及笔色的状态而定。

3. **海龟横纵向位移指令** SETXY

格式：SETXY 参数 1 参数 2

类型：语句、命令

功用：移动海龟到参数 1，参数 2 所指定的(X，Y)坐标位置。

注释：SETXY 来源于“SET X and Y coordinate”，是设置海龟的 X、Y 坐标的意思。

参数 1、2 的格式、取值范围与 SETX、SETY 相同。

说明：该命令指示海龟移位到 X 坐标为参数 1，Y 坐标为参数 2 的位置上。位移过程中，海龟尖头的方向不变，海龟所经过的轨迹是否画出，视提、落笔及笔色的状态而定。

注意：当参数 2 为负常数时，负号前面要加一个双撇号以供 LOGO 识别，例如：SETXY 30 ”−20。否则 LOGO 将把负号当做减号处理。

上述三条命令都可以令海龟在位移中画线，因此也可以使用这三条命令绘图。

4. **坐标位移指令的应用**

在实际绘图中，有些图形，计算前进步数和转角比较困难，若使用坐标定位法将很容易实现。如图 5－31 所示的三角形，只要知道各顶点的坐标，编写绘图过程将非常简单（见过程 D）。

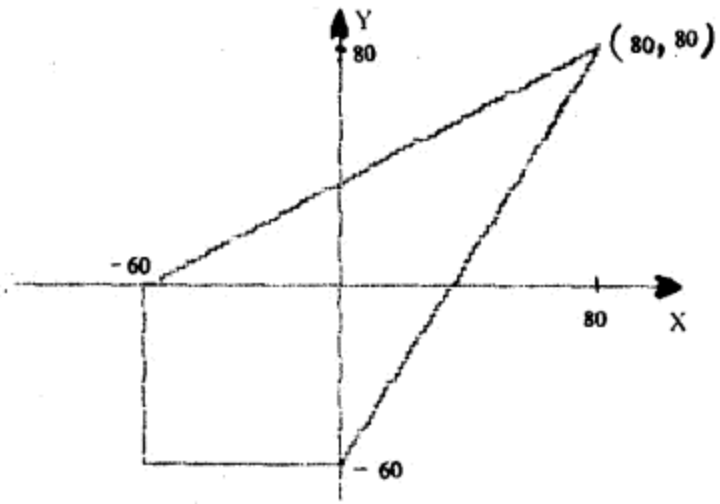


图 5－31　三角形

```
TO D
  PU SETX －60 PD SETY －60 SETX 0
  SETXY 80 80 SETXY －60 0
END
```

再如要画如图 5－32 所示的房子，画墙体、门、窗很容易，编一个过程 C 就可以画出各种长方形。但画房顶（梯形）时，计算前进步数和转角就有些困难。如果用坐标定位法画梯形就很简单。编写过程如下：

```
TO C B1 B2
  REPEAT 2 [FD :B1 RT 90 FD :B2 RT 90]
END
```

```
TO FANG
  HT PU SETXY -80 "-60 PD C 120 160
  SETX 15 C 80 35 PU SETXY -40 20
  PD REPEAT 4 [C 20 20 RT 90] PU
  SETXY -80 60 PD SETXY -60 100
  SETX 60 SETXY 80 60
END
```

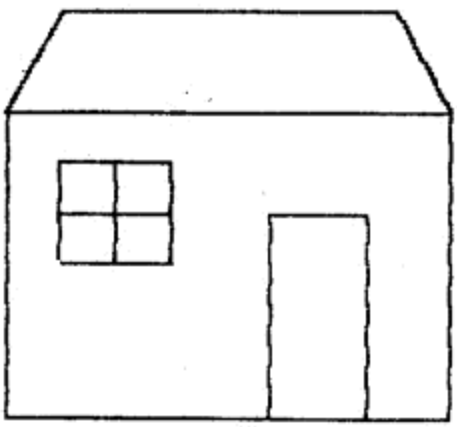


图 5－32　房子

执行过程“FANG”就可画出图 5－32 所示的房子。

四、海龟的定向

使用海龟定向指令“SETH”，可以确定海龟的方向。

格式：SETH　角度值

类型：语句、命令

功用：设置海龟尖头为规定的角度。

注释：SETH 来源于“SET Heading”，是“设置海龟尖头方向”的意思。

LOGO 规定，在绝对坐标系中，海龟尖头朝上的角度值为零度，按顺时针方向转动时角度值为正；按逆时针方向转动时

角度值为负。

说明:角度值不是海龟转动的角度数,而是转向终结时海龟尖头的绝对方向角。比如键入:

? SETH 90

不管当前海龟尖头指向哪个方向,只要按回车键,海龟尖头立即指向正右方。若执行命令:

? SETH 0

海龟头尖总是指向正上方。

五、海龟坐标的查询

建立了绝对坐标系以后,海龟在屏幕上的任一点都有一对唯一的实数值与其对应。这样以来,海龟的位置描述就有了共同语言,编程中就可以使用查询指令找出海龟的当前坐标。

1. 海龟的 X 坐标测试指令 XCOR

格式:XCOR

类型:语句、命令

功用:测试海龟当前位于屏幕 X 方向的绝对坐标值。

注释:XCOR 来源于"X axis COoRdinate",是"测试 X 轴坐标数值的意思"。

XCOR 命令返回的是海龟在 X 方向的绝对坐标值,取值范围为−128～+127。

说明:X 的绝对值是海龟从当前位置沿 X 轴方向爬到 Y 轴(屏幕中心的竖线)时所需的步数。海龟在纵轴右边,X 值为正;海龟在纵轴左边 X 值为负。

示例:键入

? XCOR

将显示

RESULT：7.44191241

表示海龟当前位置的X坐标值7.44191241。

注——XCOR输出的海龟坐标值为浮点数，但海龟绘图中按整数值确定坐标，即使使用了浮点坐标，系统会自动取整。

2. **海龟的Y坐标测试指令** YCOR

格式：YCOR

类型：语句、命令

功用：测试海龟当前位于屏幕Y方向的绝对坐标值。

注释：YCOR来源于"Y axis COoRdinate"，是"测试Y轴坐标数值的意思"。

YCOR命令返回的是海龟在Y方向的绝对坐标值，取值范围为－119～＋120。

说明：Y的绝对值是海龟从当前位置沿Y轴方向爬到X轴(屏幕中心的横线)时所需的步数。海龟在横轴上方，Y值为正；海龟在横轴下方Y值为负。

示例：键入

？YCOR ↙

显示

RESULT：28.7083893

3. **海龟指向测试指令** HEADING

格式：HEADING

类型：语句、命令

功用：测试当前海龟尖头的指向与Y轴的夹角(即在绝对坐标系中的角度)。

注释：HEADING在英文中为"指向"的意思。

说明：输出海龟当前方向的角度值，这个角度以正上方为

零度基线，顺时针方向为正，逆时针方向为负。

示例：键入

？HEADING ↙

显示

RESULT：29.9999998

近似等于30°（由于电脑采用对数运算方法，计算结果有一定的误差）。

第七节　彩色世界

到目前为止，我们所画的图形都是用白色的线条画在黑色的屏幕上，画面较单调，本节将讨论彩色绘图的方法。

一、彩色绘图的有关指令

1. **画笔选择指令** PICKPEN

格式：PICKPEN :N

类型：语句、命令

功用：选择N号画笔。

注释：PICKPEN在英文中为“拿笔”的意思。

说明：LOGO语言中有四支画笔，编号为0～3。其中0号控制画底背景，1～3号笔用来画彩色图形。

示例：

？PICKPEN 3

选择了3号画笔。

这条命令还有一个用途就是“擦除部分图形”。由于LOGO语言中，没有擦除部分图形的命令，使用底背景笔绘图就可达到擦图的目的。例如，当把画正三角形的命令误输为

？REPEAT 3 [FD 60 RT 120] LT 90 FD 60

画成了如图 5－33 所示的图形时，多出的一笔，可用 PICKPEN 命令选择 0 号画笔后退 60 步擦除。即输入

？PICKPEN 0 BK 60

回车后，就可以把多出的一笔擦掉。

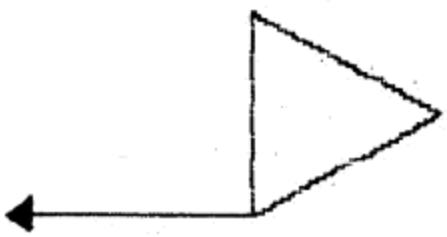


图 5－33　部分擦除示例图

2. 设置画笔颜色指令 PC

格式：PC ：N ：X

类型：语句、命令

功用：设置 N 号画笔为 X 号代码的颜色（颜色与代码的对应关系见表 5－1）。

注释：PC 来源于“Pen Color”，为“画笔的颜色”的意思。

N——画笔号，取值范围 0～3，可以取数值常数、变量或表达式的形式。

X——画笔的设定颜色代码，取值范围为 0～15，可以取数值常数、变量或表达式的形式。

示例：如欲用 2 号画笔画红色图形，可键入

？PC 2 6 REPEAT 5 [FD 60 RT 144]

回车后，就可以在黑色的屏幕上画出一个红色的五角星。

表 5—1　画笔颜色代码一览表

代码	颜色	代码	颜色	代码	颜色
0	白	6	红	12	浅蓝
1	蓝	7	橙	13	灰
2	蓝紫	8	黄	14	黑
3	紫	9	黄绿	15	透明
4	紫红	10	绿		
5	粉红	11	深绿		

3. **设置背景颜色指令** BG(BACKGROUND)

格式:BG :X

类型:语句、命令

功用:设置屏幕背景的颜色,同时设置 0 号画笔的颜色。

注释:BG 来源于"Back Ground",为"背景"的意思。

说明:X 为选定的背景颜色,取值范围为 0～15(颜色、代码对应关系见表 5—1),可以取数值常数、变量或表达式的形式。

示例:键入

? BG 1 REPEAT 9 [FD 80 RT 160]

回车后,将在蓝色的背景上画出一个白色的九角星。

注意——必须先置背景色后绘图,否则,图形的颜色将发生变化。

4. **设置颜色亮度指令** BRIGHT

格式:BRIGHT :N :X

类型:语句、命令

功用:设置 N 号笔的颜色亮度。

注释:BRIGHT 在英文中为“明亮”的意思。

N——为画笔的号码,取值范围 0~3,可取数值常数、变量或表达式的形式。N=0 时,设置背景亮度。

X——为颜色亮度代码,取值为 0、1,X=0 时为低亮度;X=1 时为高亮度。X 也可以取变量或表达式的形式。

示例:

利用色彩的变化可以使图形产生运动的效果。例如画一个飞行的火箭,过程如“S”所示。

```
TO S
  HT REPEAT 3 [FD 10 RT 120]
END

TO HJ
  LT 60 S RT 60 PU FD 10 PD FD 10
  RT 30 S RT 60 PU FD 10 PD RT 90
  FD 10 PU FD 10 PD RT 180 S LT 90
  FD 10 RT 90
END

TO FEI B
  REPEAT 20 [HJ PICKPEN 0 HJ FD :B PICKPEN
 2]
END
```

过程说明:

过程“S”用于绘制小三角形,过程“HJ”嵌套过程“S”用于绘制火箭。

“FEI”的过程体中先使用 2 号笔画火箭,后使用 0 号笔

(即画背景色),把笔划重复的图形擦掉,从而产生了火箭向前运动的效果(见图 5—34)。

过程"FEI"将演示向上飞行的火箭。由于是在允许海龟绕圈的状态下绘图,所以当火箭从上面越出屏幕后又会从下面钻出来继续飞行。变量"B"控制火箭的飞行速度,取值越大飞行速度越快。

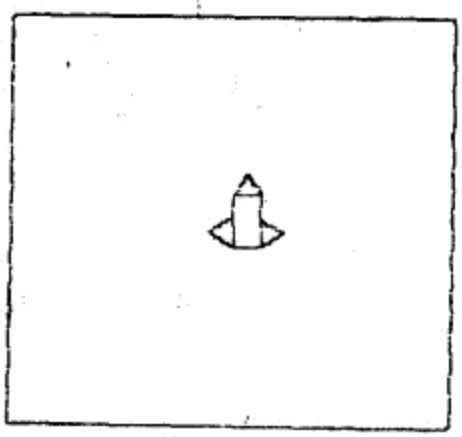


图 5—34　火箭

PDG

第六章　海龟作品

通过前面几章的学习，我们已学会了不少指挥海龟画图的命令，基本掌握了绘图过程的编写方法。为了巩固所学知识，更深入地了解各命令的功能。本章特优选了8个富有代表性的绘图过程，介绍给大家，希望能起到举一反三的作用。

生动有趣海龟作品展览馆开放了，你想参观一番吗？

第一节　直接命令绘图

一、美丽的孔雀

例6—1　画一个如图6—1所示的孔雀。

图6—1　孔雀

请大家按顺序键入下列命令：

DRAW ↙

PU LT 90 FD 50 RT 90 PD ↙

RT 20 FD 34 LT 20 ↙
REPEAT 3 [FD 10 RT 120] ↙
RT 140 FD 10 RT 120 FD 6 ↙
LT 60 FD 10 LT 60 FD 10 ↙
LT 50 FD 10 LT 30 FD 40 ↙
RT 70 FD 40 RT 140 FD 90 ↙
BK 32.5 RT 90 PU FD 5 RT 90 PD ↙
LT 30 FD 30 RT 70 FD 25 RT 140 ↙
FD 45 RT 157 FD 20 RT 70 PU FD 5 PD ↙
RT 122 FD 20 PU BK 22.5 SETH 0 FD 4 PD ↙
REPEAT 36 [FD 0.174 * 2.5 RT 10] ↙
HT ↙

现在，一只美丽可爱的孔雀呈现在你的面前。

二、矫健的雄鹰

例 6—2　画一个如图 6—2 所示的飞鹰。

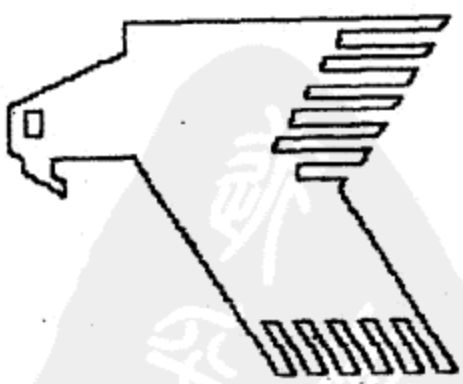


图 6—2　飞鹰

键入下列命令：

```
DRAW ↙
LT 90 FD 25 LT 90 FD 5 LT 60 FD 5 ↙
RT 60 FD 5 RT 120 FD 15 RT 60 FD 5 ↙
LT 60 FD 5 RT 60 FD 15 BK 10 RT 90 ↙
PU FD 5 LT 90 PD ↙
REPEAT 2 [FD 8 RT 90 FD 5 RT 90] ↙
LT 90 PU FD 5 RT 90 FD 10 PD RT 65 ↙
FD 40 LT 65 FD 10 RT 90 FD 100 ↙
REPEAT 5 [RT 120 FD 5 RT 60 FD 30 LT 60 FD 5
    LT 120 FD 30] ↙
RT 120 FD 5 RT 60 ↙
FD 20 LT 90 FD 5 LT 90 FD 15 RT 120 ↙
FD 5 LT 60 FD 70 RT 120 FD 5 RT 60 ↙
REPEAT 5 [FD 20 LT 60 FD 5 LT 120 FD 20 RT 120
    FD 5 RT 60] ↙
FD 85 LT 60 FD 10 ↙
HT ↙
```

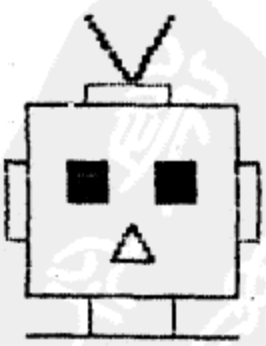


图 6—3　机器人头

三、可爱的机器人

例 6－3　画一个如图 6—3 所示的机器人头。

键入下列命令：

```
DRAW ↙
PU SETXY －10 50 PD ↙
REPEAT 2 [FD 5 RT 90 FD 20 RT 90] ↙
FD 5 SETX 0 RT 30 FD 20 BK 20 ↙
LT 60 FD 20 BK 20 RT 30 ↙
PU BK 5 RT 90 BK 25 PD ↙
REPEAT 4 [FD 50 RT 90] ↙
LT 90 BK 35 LT 90 ↙
REPEAT 2 [FD 5 RT 90 FD 20 RT 90] ↙
RT 90 FD 10 PU SETX －15 PD ↙
REPEAT 10 [FD 10 BK 10 RT 90 PU FD 1 LT 90 PD] ↙
PU SETX 6 PD ↙
REPEAT 10 [FD 10 BK 10 RT 90 PU FD 1 LT 90 PD] ↙
PU SETX 25 BK 10 PD ↙
REPEAT 2 [FD 20 RT 90 FD 5 RT 90] ↙
PU SETX －5 BK 5 PD RT 30 ↙
REPEAT 3 [FD 10 RT 120] ↙
LT 30 PU BK 20 RT 90 BK 5 PD ↙
REPEAT 2 [FD 20 LT 90 FD 10 LT 90] ↙
BK 15 FD 50 ↙
HT ↙
```

四、逗人的企鹅

例 6—4　画一个如图 6—4 所示的企鹅。

图 6—4　企鹅

键入下列命令：

DRAW ↙

HT ↙

PU FD 50 PD ↙

RT 90 FD 20 RT 90 FD 10 LT 20 FD 30 ↙

RT 20 FD 50 LT 40 FD 15 SETH 270 ↙

FD 25 RT 90 FD 35 LT 140 FD 20 RT 140 ↙

FD 30 RT 40 FD 20 LT 40 FD 15 ↙

LT 30 FD 23 LT 60 FD 17 RT 50 BK 18 ↙

RT 40 BK 12 FD 10 RT 90 PU FD 5 PD LT 180 ↙

REPEAT 36 [FD 0.174 * 3 RT 10] ↙

PU FD 5 PD RT 90 ↙

BK 15 RT 60 BK 10 LT 60 BK 40 FD 20 ↙

LT 45 FD 15 RT 45 FD 30 RT 135 ↙

FD 23 BK 23 LT 135 BK 30 LT 45 BK 15 ↙

RT 45 BK 28 LT 45 BK 15 LT 60 FD 10 ↙

LT 45 FD 10 SETH 270 BK 15 RT 120 ↙

FD 5 RT 120 FD 5 LT 60 FD 15 LT 120 FD 5 ↙

通过上面四个例题，可以体会到使用直接命令绘图有如下缺点：

1. 输入完一条命令后，只要打入回车键，就不能修改。即使存在错误也只能眼睁睁地看着海龟把画面搅的一塔糊涂。

2. 输入命令多、绘图速度慢。

3. 输入的命令无法进行保存，编程效率低。

第二节　编写过程绘图

编写过程绘图消除了直接命令绘图的弊端。一个过程被定义后，即可当做 LOGO 语言的新命令使用，还可以供其他过程调用，程序的效率比较高。

一、空心五角星

例 6—5　画一个如图 6—5 所示的空心五角星。

过程如下：

```
TO WUXING
  RT 18
  REPEAT 5 [FD 40 RT 144 FD 40 LT 72]
END
```

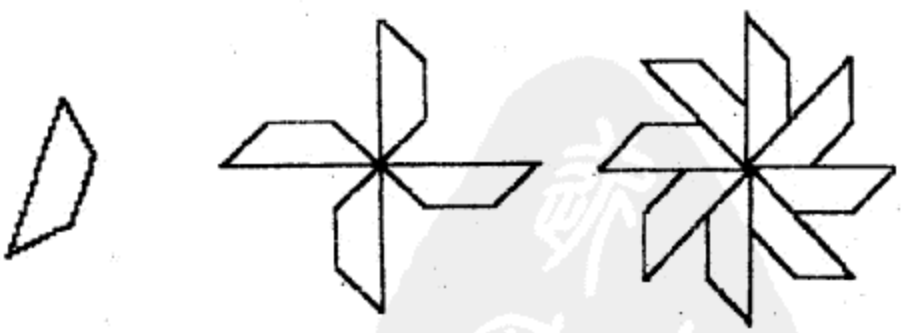
图 6—5　空心五角星

二、形象的风车

例 6—6　编写画如图 6—6 所示的风车过程。

1. 首先编一个画等腰梯形的过程“T”

```
TO T
  FD 49 RT 135 FD 20 RT 45 FD 20
  RT 45 FD 20 RT 135
END
```



(a)　(b)　(c)

图 6—6　风车

执行过程“T”将画出如图 6—6a 所示的等腰梯形。

2. 调用梯形画风车。

```
TO DT
  FULLSCREEN HT
  REPEAT 4 [T RT 90]
END
```

执行过程“DT”将画出如图 6—6b 所示的四叶风车。

3. 八页风车

如果要画如图 6—6c 所示的八叶风车，只要将“DT”过程中的“4、90”改为“8、45”即可。修改后的过程如下：

```
TO XDT
  FULLSCREEN HT
  REPEAT 8 [T RT 45]
END
```

上例中，由于风车的图形比较大，若使用混合屏，风车下部将被文本显示区遮盖，所以绘图前要先转换成图形屏。

三、令人神往的少年宫

例 6—7　画一幢如图 6—7 所示的大楼。

图 6—7　大楼

过程如下：

```
TO FZ
  PU SETX -50 PD
  HT C1 20 30 C1 30 50 C1 20 30 LT 90
  FD 110 RT 90 FD 20 RT 90 FD 10 LT 90
  C1 15 20 FD 10 C1 25 15 C1 40 20
  C1 25 15 BK 10 C1 15 20 LT 90
  PU FD 55 RT 90 FD 50 PD RT 30 FD 20
  RT 120 FD 20 BK 20 LT 150 FD 15 C 5 7
END

TO C1 S K
  C :S :K RT 90 FD :K/4 LT 90
  REPEAT 3 [PU FD :S/4 PD FD :S/2 PU FD :S/4
  BK :S RT 90 FD :K/4 LT 90 PD]
END

TO C S K
  REPEAT 2 [FD :S RT 90 FD :K RT 90]
END
```

四、晴雨伞

例 6—8 画雨伞。

上面实例绘制的图形都是直线结构的。现在介绍曲线结构图形的实例。画雨伞的过程如下：

```
TO HU S R A
  REPEAT :S/10 [RT :A FD :R * 0.174 RT :A]
```

```
END

TO SAN
  FULLSCREEN
  HU 90 30 5
  LT 90 FD 5 BK 5 RT 90
  HU 90 30 5 RT 180
  REPEAT 3 [HU 180 (-10) (-5) RT 180]
  HU 180 (-10) 5 RT 180 HU 90 10 5
  RT 90 FD 50 HU 180 5 5
  HT
END
```

执行“SAN”将画出图 6—8 所示的图形。

图 6—8　雨伞

一般情况下，曲线绘图要准备四个画弧的过程，即右转前进弧、右转后退弧、左转前进弧、左转后退弧。绘制图 6—8 的过程，通常应当使用三种画弧过程，但在过程“SAN ”中只用了一个画弧过程，这里体现了编程的技巧性。对过程不作详细讨论，留给读者分析，下面仅说明变量的作用供分析过程时参

考。

变量“S”表示圆所对应的圆心角。“R”表示圆弧的半径，取值大于零表示前进弧；取值小于零表示后退弧。“A ”表示海龟每次转的角度，规定取值为正时海龟向右转，取值为负时海龟向左转。

许多海龟图，乍看上去好象很复杂，其绘图过程则往往比较简单。这是过程反复优化的结晶。要想熟练地、简洁地编写各种海龟图的绘图过程，只有善于把复杂图形分解为基本的、简单图形的组合，善于抽取复杂图形的基本成分，善于运用移动、旋转、重复、调用等工作方式，才能绘出富有创意的图形。

第七章　递归过程

通过以上几章的学习，我们对过程的定义和调用已十分熟悉。本章将讨论过程的另一种使用方法——过程的递归调用。

过程的递归调用又称“递归过程”，它是过程自己调用自己的一种调用形式。递归过程是 LOGO 语言的难点，尤其是中间递归和多层递归最不容易掌握。希望大家在“走程序”上多下功夫(就是跟着程序一步一步地往下走)，弄清楚每一步要做什么，怎样运行、画出什么样的图形。

结合递归过程的讨论，本章将介绍下列指令：

逻辑非运算指令　NOT
逻辑与运算指令　ALLOF
逻辑或运算指令　ANYOF
条件转移指令　IF……THEN
　　IF……THEN ELSE
无条件转移指令　GO
退出过程指令　STOP
返回命令状态指令 TOPLEVEL

第一节　递归与递归过程

一、递归的概念

下面仅列举两例说明“递归”的概念。

1. 从前有座山

例 7－1　有这样一个故事：

“从前有座山，山里有座庙，庙里有个老和尚正在讲故事，讲的是什么呢？”

“从前有座山，山里有座庙，庙里有个老和尚正在讲故事，讲的是什么呢？”

“从前有座山……”。

这种故事里面讲述故事本身的形式，就是“故事递归”的概念。

一种事情的内容中包含了这件事情本身就是“递归”。

2. 算苹果

例 7－2　某人到外地打工，回家时买了一筐苹果。他先探望了五家亲戚，每走一家亲戚，就将苹果放下一半，待回到自己家时只剩下三个苹果。问这个人一共买了多少个苹果？

分析问题

为了计算共买了多少个苹果，就必须知道去第一个亲戚家时放下苹果的个数，要求第一家放的个数，又必须知道第二家放的个数，要求第二家放的个数，还必须知道第三家放的个数……，依次类推，欲求第五家放的个数就必须知道最后剩了多少个苹果。

设：共买了 PG0 个苹果，第一家放的个数为 PG1、第二家

放的个数为 PG2、……、最后回到自己家时剩下的苹果数为 PG6。

那么可以写出如下关系式：

PG0＝PG1×2

PG1＝PG2×2

PG2＝PG3×2

PG3＝PG4×2

PG4＝PG5×2

PG5＝PG6×2

因为 PG6＝3，所以有

PG5＝3×2＝6

PG4＝6×2＝12

……

PG0＝PG1×2＝96×2＝192 个

归纳起来，计算苹果总数的算式为：

PG0 ＝PG6×2×2×2×2×2×2＝3×2×2×2×2×2×2

＝192 个。

这种把求未知结果回归成对已知内容进行求解的过程，就叫做“递归”。

通过上述的示例，你可能已初步认识“递归”了。

递归——从 LOGO 的角度讲，就是指一个过程直接或间接地调用它本身，这种调用方法叫做“递归调用”，相应的过程就叫做“递归过程”。

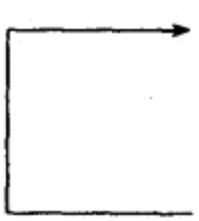
二、递归过程的概念

在过程体中，出现本过程的过程名的过程，就是递归过程。最简单的递归过程形式是“尾递归”。“尾递归”的语句形

式就是在过程体的最后一句使用递归。

为了说明递归的概念，让我们先看一个画圆过程的语句形式。前面，我们曾讨论过用重复命令画圆的方法，现在我们采用“尾递归”来完成画圆。

画圆的“尾递归”过程如下：



```
TO Y
RT 10 FD 10
Y
END
```

这个过程极其简单，键入“Y”，回车后，将看到海龟画出一个圆形。但是海龟画完圆后，并不停止，仍然沿着圆的轨迹继续画啊……、画啊……，永不停止，只有按 BREAK 键(PAUSE 或 STOP 键)海龟才会停下来。这种重复执行某段程序的现象通常称为“死循环”。

三、递归过程工作原理分析

现在，我们来分析一下这个过程的执行经过。执行过程后，“RT 10”控制海龟右转 10°，然后执行“FD 10 ”令海龟前进 10 步，海龟画出一小段圆弧；最后遇到本过程的过程名“Y”，就返回过程头再执行过程体，又画一小段圆弧；又回到过程头……。这样循环 36 次便画出一个圆。画完一个圆后，仍然返回过程头继续画圆弧，此后将沿着第一个圆的轨迹画出第二、第三……个圆。由于过程中没有停止命令，它将“行行复行行”，一圈一圈地、无休止地画下去，陷入死循环状态。显然，只有打断程序的运行，海龟才会停下来。按“BREAK”键

后，屏幕显示

STOP!

的信息，表示过程的运行被强迫终止。

四、带参数递归过程

下面讨论稍微复杂一点的递归——带参数的递归过程。

请看下面的例子：

例 7—3　多角星的演绎。

过程见“DT”。

```
TO  DT  B  J
  FD  :B  RT  :J
  DT  :B  :J
END
```

在过程“DT”中，变量“B”用来控制海龟前进的步数；变量“J”用来控制海龟转弯的角度数。在过程体的尾部，进行了递归调用。这个过程同前面画圆的过程基本相似，都是在过程的尾部出现过程名，形成递归。所不同的是它连同参数一起进行递归。由于参数可以赋予不同的值，所以，可以利用带参数的过程画出不同的图形。

现在让我们执行“DT”过程看一看所画图形的效果。

当键入：

? DT 70 250

回车后，将画出如图 7—1a 所示的图形；

当键入：

? DT 60 160

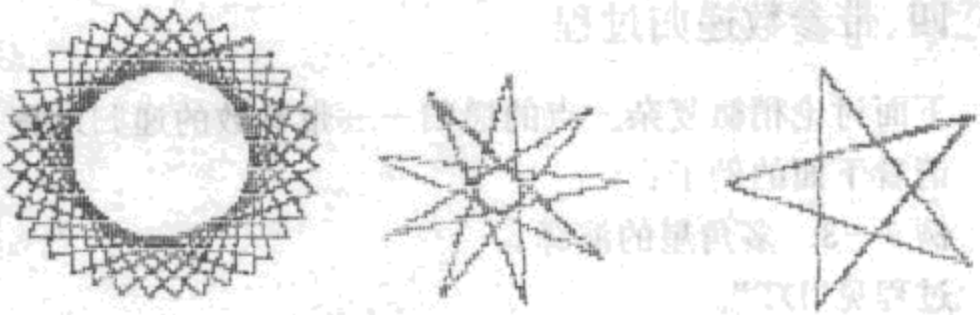
回车后，将画出如图 7—1b 所示的图形；

当键入：

? DT 70 144

时，将画出如图 7－1C 所示的图形。

以上两例说明了递归过程可使程序结构得到高度优化。



(a)　　(b)　　(c)

图 7－1　多角星的演绎

在定义带参数的递归过程时，需要特别注意被调用同名过程的参数个数，必须与过程首部的参数个数相同。

五、带参数递归的应用

带参数递归过程最擅长的手段，是绘制各种螺旋曲线。下面仅示两例。

1. 转角不变的螺旋线

例 7－4　画旋涡线。

过程如下：

```
TO XX B J L
  HT FD :B RT :J
  XX :B+:L :J :L
END
```

过程中参数“B”是每次画图的边长，参数“L”是每递归调用一次“B”的增量，参数“J”是转向角度。

执行这个过程时，要分别给三个参数赋初值：边长可以从任意的初始值开始（一般从最小边长开始向外画，取零值）；递归步长（每次递归边长的增加值）任意，步长越小螺旋线越密；转角可分为两类，一类是某种多边形的转角，这样将画出正规的多边形螺旋线，另一类是接近多边形或多角星的值，这样画出的螺旋线近似多边形或多角星。

键入：

? XX 0 90 3

回车后，将画出如图 7－2a 所示的方形螺旋线。

键入：

? XX 0 1180 2

回车后，将画出如图 7－2b 所示的变形三角螺旋线。

键入：

? XX 0 45 0.5

回车后，将画出如图 7－2c 所示的八边形螺旋线。

在上述过程中，若再加入彩色，绘出的图形效果将更加丰富多彩。

键入：

? PC 3 6 XX 0 144 0.4

回车后，将画出如图 7－2d 所示的红色五角星。

图 7－2 的 4 个图形是在角度值不变，仅改变前进步数的情况下绘出的。现在我们固定前进步数不变，仅改变角度值，看看能绘出什么样的图形。

2. 边长不变的螺旋线

例 7－5　修改过程“XX”为“JX”，观察绘图效果。

```
TO JX :B :J :L
  HT FD :B RT :J
```

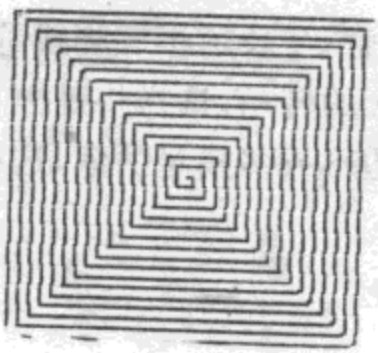


图 7－2a　方形螺旋线

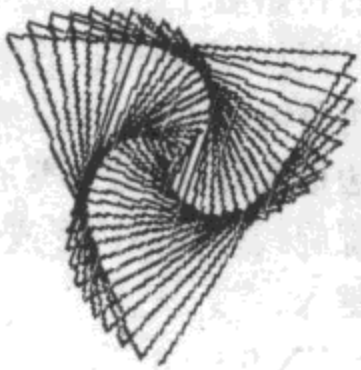


图 7－2b　变形三角螺旋线

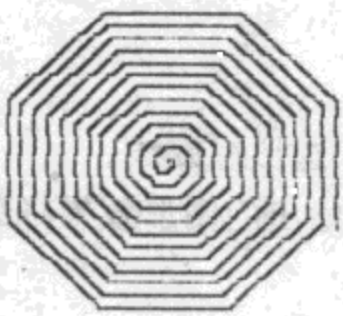


图 7－2c　八边螺旋线

图 7－2d　五角星

```
    JX :B :J+:L :L
END
```

执行“JX”过程，当键入：

```
? JX 8 3 10
```

时，将绘出如图 7－3a 所示的图形。

当键入：

```
? JX 15 1 10
```

时，将绘出如图 7－3b 所示的图形。

当键入：

？ JX 13 2 10

时，将绘出如图 7－3c 所示的图形。

当键入：

？ JX 10 3 100

时，将绘出如图 7－3d 所示的图形。

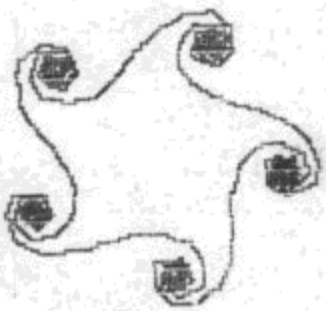


图 7－3a

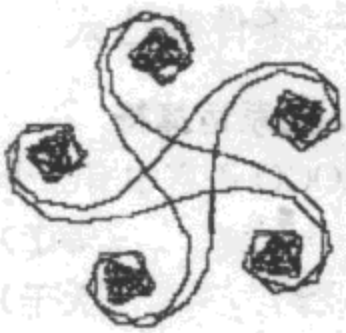


图 7－3b

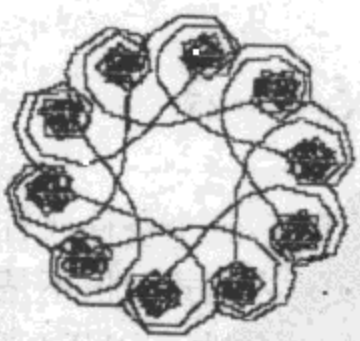


图 7－3c

图 7－3d

真有趣，如此简单的过程竟能画出如此美妙、神奇的图形，简直不可思义。

相信你已对递归产生兴趣了。

第二节　条件转移

上一节的递归过程执行中，遇到了无限递归循环的问题。本节将讨论如何退出递归的方法。

一、条件表达式

1. 关系运算符

LOGO 语言中有三个关系运算符：

＝　　（等于）

＞　　（大于）

＜　　（小于）

2. 关系表达式

关系表达式就是用关系运算符连接起来的式子。

例如，:X＞50　表示　X＞50；

:Y＜4　表示　Y＜4；

:X＋:Y＝25　　表示 X＋Y＝25。

3. 关系表达式的值

关系表达式实际上是一个判断条件，这个条件或者成立，或者不成立，两者必居其一。因此，关系表达式的值不是数值，而是逻辑值，当关系表达式成立时，其值为 TRUE(真)，反之，其值为 FALSE(假)。例如：键入

? PR　5＞4

回车后，将显示：

TRUE

表明关系表达式 5>4 是成立的，其值为真(TRUE)。

下面的条件表达式都是成立的：

10=5*2　　7<8　　3+5>8-1

其结果都为真(TRUE)。

又如：键入

? PR 6<5

显示：

FALSE

表明关系表达式 6<5 是不成立的，其值为假(FALSE)。

下面的条件表达式都是不成立的：

2>-1　　3<0　　5*3=5+3

其结果都为假(FALSE)。

4. 逻辑运算符

LOGO 语言中有三种逻辑运算符：

NOT　　(逻辑非)

ALLOF　　(逻辑与)

ANYOF　　(逻辑或)

(1)逻辑非(NOT)

格式：NOT 条件

类型：算符

功用：输出条件的非。

注释：NOT 在英文中为"不是"的意思。

说明：条件为真时，输出则为假；条件为假时，输出则为真。NOT 逻辑运算的值见表 7-1。

表 7－1　NOT 逻辑真值表

条　件	NOT 条件
真	假
假	真

示例：

NOT ：B＝0　表示 B≠0

NOT ：B＞0　表示 B≤0

NOT ：B＜0　表示 B≥0

(2)逻辑与(ALLOF)

格式：ALLOF（条件表达式 1)(条件表达式 2)

类型：算符

功用：如果条件表达式均为“真”(TRUE)，则输出为“真”(TRUE)；如果条件表达式有一个为“假”(FALSE)，则输出为“假”(FALSE)。

注释：ALLOF 在英文中为“所有”的意思。在此表示必须所有的条件表达式都成立才行。

说明：ALLOF 总条件成立的原则是，后面所有的条件都成立时总的条件才成立，若其中有一个条件不成立则总条件不成立。

ALLOF 运算的值见表 7－2。

表 7－2　ALLOF 逻辑真值表

条件 1	条件 2	ALLOF(条件 1)(条件 2)
真	真	真
真	假	假
假	真	假
假	假	假

示例：

(A) ALLOF (2>−1) (3<0)

两个条件都不成立，总条件也不成立。

(B) ALLOF (;X>0) (;X<1)

意思是只有当 X>0 和 X<1 同时成立时总条件才成立。显然，当 X=0.5 时总条件成立；当 X=2 时总条件不成立。

(3)逻辑或(ANYOF)

格式：ANYOF (条件表达式 1)(条件表达式 2)

类型：算符

功能：如果条件表达式中有一个为真(TRUE)，则输出为真(TRUE)；如果条件表达式都为假(FALSE)，则输出为假(FALSE)。

注释：ANYOF 在英文中为“任何一个”的意思。在此表示所有条件中任何一个成立即可。

说明：ANYOF 总条件成立的原则是，后面的条件有一个成立时总的条件就成立，所有的条件都不成立时总的条件才不成立。

ANYOF 逻辑运算的值见表 7−3。

表 7−3 ANYOF 逻辑真值表

条件 1	条件 2	ANYOF(条件 1)(条件 2)
真	真	真
真	假	真
假	真	真
假	假	假

示例：

(A) ANYOF (2<−1) (3<0)

两个条件都不成立，所以总条件不成立。

(B) ANYOF (:X<0) (:X>0)

意思是：X<0 和：X>1 有一个条件成立时总的条件就成立。显然仅有：X=0 时总条件不成立。

5. 逻辑表达式

逻辑表达式是用逻辑运算符连接起来的有意义的式子，上面逻辑运算符介绍中的几个示例，就是逻辑表达式。

(1)逻辑表达式的值

逻辑表达式的值也是一个逻辑值。当逻辑表达式成立时，其值为真，反之，其值为假。

下面请看几个具体的例子。键入：

? PR NOT 3>4

显示：

TRUE　　(说明 3 不大于 4)

键入：

? ALLOF (3<4)(4<5)

显示：

TRUE　　(说明 3 小于 4、4 小于 5)

键入：

? ANYOF (4<3)(5<4)

显示：

FALSE　　(说明 4 不小于 3、5 不小于 4)

注意——逻辑与(ALLOF)和逻辑或(ANYOF)后面的条件表达式必须用圆括号括起。

6. 运算优先级

(1)表达式的优先级

关系表达式的运算优先级高于逻辑表达式，先求关系表

达式的值，然后再求逻辑表达式的值。

(2)逻辑运算符的优先级

逻辑运算符的运算优先次序为：

NOT

ALLOF

ANYOF

二、条件转移指令

上面所讨论的关系、逻辑表达式，都可用于表述一定的条件，因此可把它们统称为条件表达式。

使用条件表达式可以构成条件语句。通过条件语句的控制就可以设置递归过程调用的转出口，达到灵活使用递归过程的目的。

条件语句由条件转移指令和条件表达式组成。

格式 1：IF〈条件表达式〉THEN〈语句〉

格式 2：IF〈条件表达式〉THEN〈语句 1〉

ELSE〈语句 2〉

类型：语句、命令

功用：根据条件表达式是否成立，控制程序的走向。第一种语句形式：

当条件表达式成立时(即为真“TRUE”)则执行 THEN 后面的语句，若不成立(为假“FALSE”)则跳过该条语句，执行下一个语句。该种语句形式的“THEN”可以省略。

第二种语句形式：

当条件表达式的值为真时，执行“THEN”后面的〈语句 1〉，跳过〈语句 2〉；否则，跳过〈语句 1〉，执行“ELSE”后面的〈语句 2〉。

注释:IF……THEN 是“如果……则”的意思。

IF……THEN ELSE 是“如果……则……，否则……”的意思。

条件表达式是由关系运算符或逻辑运算符连接起来的表达式。

〈语句 1〉、〈语句 2〉可由一系列命令组成。

示例:

```
IF:A>0 THEN FD 50
BK 30
```

这里的“:A>0”是条件表达式，当其值为真时，先执行命令“FD 50”，再执行“BK 30”；否则仅执行“BK 30”。

上述命令也可以写成:

```
IF :A>0 THEN FD 50 ELSE BK 30
```

三、过程运行的退出

1. 退出过程指令 STOP

格式:STOP

类型:语句

注释:STOP 在英文中为“停止”的意思。

功用:停止当前过程的执行，返回到调用当前过程的过程。

说明:该指令只能用于过程中，在命令状态下使用无效。当过程执行到“STOP”语句时，就会停止它所在过程的执行，返回到调用这个过程的过程中去继续执行。若仅有一个过程则返回命令状态。

示例:

```
TO A
```

```
  REPEAT 4 [FD 30 RT 90]
  STOP
  REPEAT 3 [FD 30 RT 120]
END
```

执行过程“A”后，屏幕显示出边长为 30 的正方形，立即停止过程“A”的执行，返回到命令状态。又如：

```
TO B
  REPEAT 5 [FD 30 RT 360/5]
  A
  REPEAT 6 [FD 30 RT 60]
END
```

其中“A”为第一个过程。

执行过程“B”时，海龟依次画出边长为 30 的正五边形、正方形和正六边形。这是因为当执行过程“B”后，首先执行第一行语句，画一个正五边形；然后执行第二行语句，调用过程“A”，画出一个正方形后，当执行到“STOP”命令时，停止过程“A”的执行，返回到过程“B”中继续执行第三行语句，画出正六边形。

2. **返回命令状态指令** TOPLEVEL

格式：TOPLEVEL

类型：语句

功用：停止当前过程的执行，直接返回命令状态。

注释：TOPLEVEL 英文直译为“最顶级”，在此表示“退出所有的过程，返回命令状态”的意思。

说明：TOPLEVEL 与 STOP 的不同是，它不管当前处于哪一级过程调用，都将退出过程运行，返回命令状态。

示例：

```
TO IN A
  IF :A=10 TOPLEVEL
  PR :A
  IN :A+1
END
```

键入

```
? IN 7
```

将显示

```
7
8
9
```

当参数 A=10 时，结束过程的执行，返回命令状态。

四、条件转移语句的应用

灵活使用条件语句和过程退出语句，就可以方便地控制递归过程的转移，退出递归死循环的状态。

下面讨论如何控制递归调用的退出。

1. 递归退出过程的编写

在画螺旋线的过程中，如果要求边长超过 100 时停止绘图，可将过程修改如下：

```
TO XX B J L
  IF :B>100 THEN STOP
  HT FD :B RT :J
  XX :B+:L :J :L
END
```

过程体的第一行就是递归调用的退出语句，控制条件是：如果 B>100 则执行“STOP”命令，返回其上一级的调用过

程。

2. **递归退出分析**

现在分析执行

? XX 0 90 2

的情况。

(1)执行 XX 0 90 2,三个参数的初值依次为

B=0　　J=90　　L=2

此时条件语句中的条件:B>100 不成立(为假),跳过该行语句。

(2)向下执行 FD 0 RT 90,即转了一个 90 度角,然后是一级递归调用,令:B+2,有:B=2。

(3)进行第二次条件判断,条件 B>100 仍不成立,执行 FD 2 RT 90,画出了一条长为 2 海龟步的线条。

(4)接着进行二级、三级……递归调用。直到第五十次递归调用时,参数 B=102,条件表达式:B>100 的值为真,执行 STOP 命令,结束了第五十级递归调用返回到第四十九级递归调用的过程。继续执行 XX 102 90 2 的下一条命令,也就是 END,于是结束了第四十九级递归调用,返回到四十八级递归调用的过程。继续执行 XX 100 90 2 的下一条命令,即 END。如此逐级返回,在返回过程中,由于返回之处都是 END 命令,所以每返回一级就结束了这级的递归调用,一直返回到命令状态,也就结束了过程的执行,绘图效果如图 7—4 所示。

3. **方块**

例 7—6　编写画实心方块的过程。

```
TO FK B
  HT IF :B>15 STOP
```

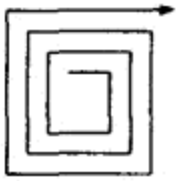


图7－4　螺旋线

```
REPEAT 4 [FD :B RT 90]
FK :B+1
END
```

执行“FK 0”将画出如图 7－5a 所示的实心方块。绘图过程是：执行一次过程体画一个正方形，每画一次后，边长加 1，由于每次边长只加 1，两次画的笔划就挨在一起，所以画出来的图形就是一个实心图。当 B>15 时停止绘图。这样就画出了一个边长为 15 的实心方块。

图 7－5a　实心方块

图 7－5b　空心方块

若执行“FK 15 ”就可以画出一个边长为 15 的空心方块（见图 7－5b）。绘图过程是：由于一开始参数 B 就等于 15，所以只执行一次过程体，就满足了递归退出条件。

可见，过程“FK”既能画实心方块又能画空心方块。是一个画方块的通用过程。

4. **棋盘**

调用过程“FK”，可以画出国际象棋的棋盘。过程如下：

```
TO QP :Y
  IF :Y>4 STOP
  REPEAT 4 [FK 15 FD 15 FK 1 FD 15]
  RT 90 FD 30 RT 90
  REPEAT 4 [FK 15 FD 15 FK 1 FD 15]
  RT 180
  QP :Y+1
END
```

执行过程“QP 1”便可画出图 7－6 所示棋盘。

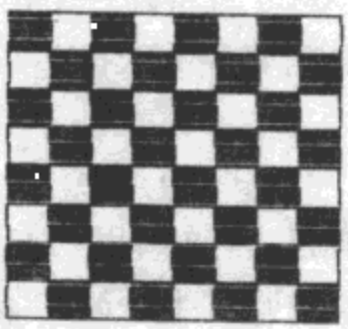


图 7－6　国际象棋盘

绘图过程是：重复语句中，每一次都画出一个空心一个实心上下相连的两个方块，重复 4 次就画出一列方块。然后把海龟移到第二列的右上角，从上往下再画一列方块。由参数 Y 控制递归调用“QP”的次数，执行 4 次过程体后，就画好了一张棋盘。

5. **实心圆环**

例 7－7　编写画实心圆环的过程。

```
TO YH X R
```

```
  HT REPEAT :X [RT 10 FD 0.174 * 2 * :R RT 10]
END

TO SY R R1
  IF (:R<:R1) STOP
  YH 18 :R RT 90 FD 1 LT 90
  SY :R-1 :R1
END
```

过程说明：

过程“YH”用于画圆弧，其中参数 X 控制重复次数，X＝18 时将画一个整圆，X＝9 时将画一个半圆。

过程“SY”用于画内半径为 R1 的实心圆环，参数 R 的初值是圆环的外半径。

绘图过程：

从最外边的圆画起，首先调用“YH”过程画一个最大的圆，然后向内走一步，递归时半径减 1，画一个紧挨着上一个圆的圆，一直画到半径为 R1 的圆时为止，这样就画出一个实心圆环。

有了画实心圆环的过程，就可以利用它画出更美妙的图形。现在我们调用这个过程画一个靶子。过程如“BA”：

```
TO BA R
  IF :R<1 STOP
  SY :R :R-10 PU RT 90 FD 20 PD LT 90
  BA :R-30
END
```

执行“BA 70”就可画出如图 7－7 所示的靶子。

图 7—7　靶子

6. **熊猫**

利用“YH”过程还可以画出逗人的熊猫不倒翁。

例 7—8　编写画熊猫不倒翁的过程。

首先编写一个画实心半圆的过程“BY”和坐标定位的过程“XY”。

```
TO BY R
  IF :R<2 STOP
  YH 9 :R RT 90 FD :R * 2 RT 90
  BY :R-1
END

TO XY X Y
  PU SETXY :X :Y PD
END
```

调用过程“XY”时，只要给出目标的坐标，就可以将海龟移到指定的位置。

然后编写画熊猫的过程“XM”。

```
TO XM
  XY -50 50 LT 10 YH 2 50 LT 68
  BY 20 RT 70 YH 4 50 LT 68 BY 20
  RT 70 YH 12 50 XY -40 60 SY 15 5
  XY 10 60 SY 15 5 XY 10 40
  SETH 180 BY 10 XY 25 15 SETH 120
  YH 3 55 SETH 180 BY 53 RT 90
  FD 100 RT 90 YH 3 55
END
```

熊猫过程“XY”嵌套了过程“BY、XY”和例 7－7 中的过程“YH、SY”。执行“XM”过程就可以画出图 7－8 所示的图形。

图 7－8　熊猫不倒翁

你能将画熊猫的整个程序走下来吗？

提示：本程序共用了五个过程。在走程序时，请注意条件语句和 STOP 语句的作用。

第三节　无条件转移

LOGO 的无条件转移指令为“GO”，使用方法如下。

格式：GO ″目标行号

类型：语句

功用：无条件的跳转到所指定的命令行去执行。

注释：GO 在英文中为“走”的意思。在此表示“走到目标行”。

目标行号为某一语句行的标号。标号可以由字母、数字组成，标号后面加冒号予以标志。

说明：这个命令只能用在过程编辑状态，不能在命令状态下执行。

示例：

```
TO AA
  DRAW
  HT
  B: FD 5 RT 5
  GO ″B
END
```

执行过程“AA”将画出一个圆。

注意：标号后面一定要带有冒号，这个冒号是提供 LOGO 识别标号的标志，它必须紧跟在标号的后面，标号与冒号之间不允许有空格，冒号与后面的命令之间必须有空格。

因“GO”语句的效率较低，一般仅在不得已的情况下才使用它，通常多用递归代替。读者只要知道这个概念就可以了。

第四节　中间递归

一、递归的分类

为了进一步研究递归过程，人们根据递归调用的位置和次数，将递归过程分为：

尾递归

中间递归

多层递归

三大类。

1. 凡递归调用在过程体的末尾处(END 命令之前)，称为“尾递归”，相应的过程称为“尾递归过程”。

2. 凡递归过程出现在过程体中间且不是最后一行的递归称为“中间递归”，这种递归调用称为“中间递归调用”，相应的过程称为“中间递归过程”。

3. 在过程体中有两次或多次出现本过程的过程名的递归称为“多重递归”，相应地也有“多重递归调用”和“多重递归过程”。

上两节我们已讨论了尾递归调用和尾递归过程，本节将重点介绍中间递归，多重递归将在下一节介绍。

二、中间递归示例

中间递归的运行情况比较复杂，为了摸清楚它的规律，我们先来看几个简单的例题。

例 7—9 简单的中间递归过程(1)。

```
TO L1 X
```

```
  IF :X=0 THEN STOP
  FD 70 RT 90
  L1 :X-1
  RT 60 FD 30
END
```

这就是一个简单的中间递归过程。执行“L1 3”将画出如图 7—9 所示的图形

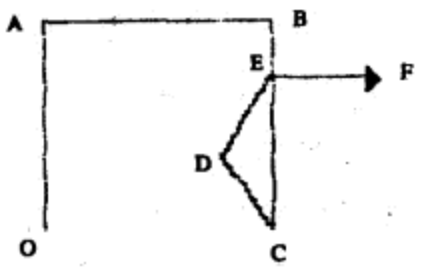


图 7—9

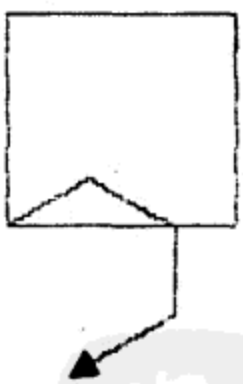


图 7—10

下面我们来分析这个过程的运行情况：

(1)海龟开始位置在 0 点，也就是母位。

(2)在执行过程“L1”时给变量 X 一个初始值 3；执行过

程体的第一句，由于 X＝3 不满足 X＝0 的条件，则转去执行过程体的第二行“FD 70 RT 90”，使海龟移到 A 点，方向向右；下面遇到 L1，递归，回到过程头，使 X－1。

(3)第二次执行过程体，这时 X＝2，还不满足 X＝0 的条件，所以又继续执行“FD 70 RT 90”，使海龟移到 B 点，方向向下；，又遇到 L1，递归，回到过程头，使 X－1。

(4)第三次执行过程体，这时 X＝1，还不满足 X＝0 的条件，所以又继续执行“FD 70 RT 90”，使海龟移到 C 点，方向向左，又遇到 L1，递归，回到过程头，使 X－1。

(5)第四次执行过程体，这时 X＝0，满足了 X＝0 的条件，执行 STOP 命令，递归结束。

(6)开始执行递归后面的命令，但不是只执行一次。由于前面进行了三次递归调用，因此后面的命令也要执行三次才能逐次退出递归。执行三次“RT 60 FD 30”，使海龟依次走到 D、E、F 点停止，海龟尖头向上。

如果执行“L1 4”，就会画出图 7－10 的图形，你能知道这是为什么吗？请动脑筋想一想。

例 7－10　简单的中间递归过程(2)。

```
TO L2 B
  IF :B<5 STOP
  REPEAT 2 [FD :B/2 RT 90 FD :B/2]
  RT 180
  L2 :B/2
  REPEAT 2 [BK B:/2 RT 90 BK :B/2]
  LT 180
  HT
END
```

请键入：

LT 90

LT 60

将画出图 7－11 所示的图形。你能根据执行的结果，分析出执行“L2 40”的情况吗？（执行情况的分析请见本章第六节）

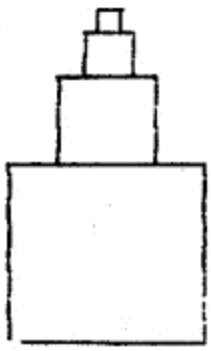


图 7－11

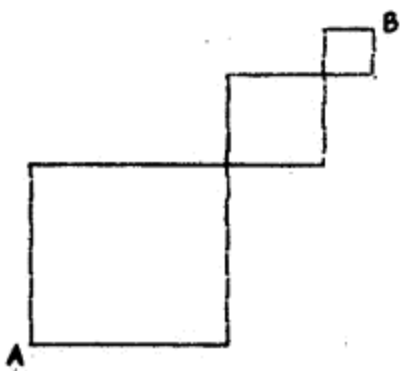


图 7－12

例 7－11　简单的中间递归过程(3)

```
TO L3 B
  IF :B<10 STOP
  FD :B RT 90 FD :B LT 90
  L3 :B/2
```

```
    RT 180 FD :B RT 90 FD :B RT 90
END
```

执行“L3 60”将画出图 7－12 所示的图形。

过程“L3”进行了三次递归，每次边长减半，B＜10 时递归结束。

执行过程“L3”，令海龟从 A 点开始沿图形的左边移动画到 B 点。递归以后的命令仍要执行三次（与递归次数相同的次数），而且把三次递归、每次减半的过程翻转返回，每退一层，恢复本次递归时的边长值。这样又从 B 点沿图形的右下沿返回到 A 点，从而画出三个相连的方块。

在编写过程画图时，可以用不同的方法来完成同一个图形的绘制。例如要画图 7－12 的图形，可以用例 7－11 的过程，也可以编写尾递归过程来完成。

将过程“L3”改为：

```
TO L. 3 B
    HT IF :B＜10 STOP
    REPEAT 6 [FD :B RT 90] RT 180
    L. 3 :B/2
END
```

执行“L. 3”，同样画出图 7－12 图形。这表明：解决同样一个问题，可以有多种方法。

在“L3”的基础上，可以再编写一个嵌套过程“L4”，将画出更有趣的图形。

```
TO L4 B
  HT REPEAT 4 [FD :B L3 :B/2]
END
```

执行“L4 60”，将画出四个如图 7－12 所示图形连接在

一起的图形(见图 7－13)。

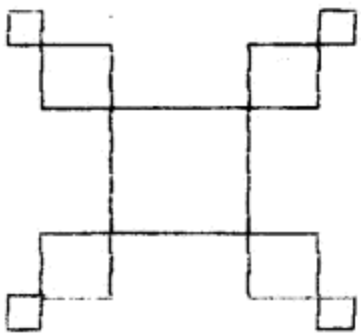


图 7－13

总结以上中间递归过程的执行情况，有两点特提请大家注意：

1. 递归后面的命令的执行次数与递归的次数相同。

2. 如果递归后面的命令中有递归时改变的变量，这些命令在重复执行时，变量要按递归时变化的规律倒回去，最终回到初始值。

第五节　多重递归

在 LOGO 语言中，多重递归是一个非常有用的工具，利用多重递归，一个很短小的程序就可画出非常复杂的图形。但是，多重递归的运行经过相当，掌握难度大，是 LOGO 编程的难点，希望大家多下些功夫。

一、多重递归的过程结构

下面仅以“两杈树”绘制过程“LCS”为例，说明多重递归

的结构。

过程清单如下：

```
TO LCS L
  IF :L<10 STOP
  FD :L LT 45
  LCS :L/2
  RT 90
  LCS :L/2
  LT 45 BK :L
END
```

过程体中的第 3 行有一次递归调用，第 5 行又有一次递归调用，这是两重递归的典型结构。

执行“LCS 60”将画出一棵两杈树（见图 7—14）。

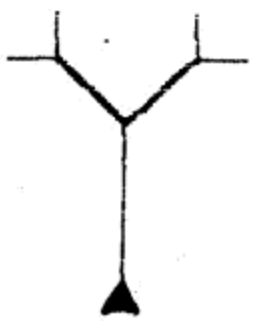


图 7—14　两杈树

二、多重递归的执行过程分析

二次递归过程的执行情况比一次递归过程复杂得多，初学者乍“走程序”（手动跟踪程序的运行过程）往往迷路，分不清走向。即使是较熟练的程序员，稍有不慎也会误入歧途，不

能回家。因此，跟踪多重递归过程的运行是奠定基本功的一个有效手段，是真正掌握LOGO语言的重要途径。当然，对LOGO语言不求甚解，只满足于一般能够编程的朋友，也可不必理会程序运行跟踪过程，跳过本段内容的叙述。

1. 递归过程中的系统内部操作简述

跟踪讨论之前，首先了解递归过程的执行中，系统内部的基本操作，将有助于过程分析的理解。

系统在执行递归过程时，是一层层的进入，再一层层退出的。每进入一层将把当前的前进参数压栈保存，每退出一层又把进栈前的参数弹出，退到最后一层时，压栈的内容全部弹出，这时递归执行过程结束（“栈”是计算机理论中的一个术语，它相当于日常生活中的“仓库”，在电脑中就是一段连续的内存区。当电脑转去作其它操作时，将把当前的有关内容保存在仓库中，术语称为“压栈”；当返回后将把以前压栈的内容从栈中取出来，术语中称为“出栈”或从栈中“弹出”。压栈和出栈遵循“先进后出、后进先出”的原则，这就如同在一座顺序存放货物的仓库中，先入库的货物被后入库的货物挡住，必须在后入库的货物先出库后，先入库的货物才能出库一样）。

2. 过程“LCS”的讨论形式

下面以执行“LCS 60”为例，讨论二重递归的操作步骤。为便于分析叙述，我们把过程“LCS ”过程体中每行语句前都加上标号，改为如下形式：

```
      TO LCS L
1       IF :L<10 STOP
2       FD :L LT 45
3       LCS :L/2
4       RT 90
```

```
5       LCS :L/2
6       LT 45 BK :L
7    END
```

注——上述过程形式仅用于讨论。

3. 过程分析叙述中的有关约定

为了使读者看清递归中各层的执行情况，特以表格的形式列出各个阶段的操作、变量取值、海龟方向等有关信息（请见表 7－4）。表中“递归深度”表示递归执行的层数，由于递归调用时参数“L”的取值减半，所以每层的 L 值为上一层的二分之一；“行号”，表示当前所执行过程体的语句行号，由它记忆着每个递归层所执行的进度。

为叙述方便，我们对图 7－14 的两杈树的各部分进行命名，从下向上、从左往右依次为：主干、左干（左边的树杈）、右干、左左枝（左干左侧的树枝）、左右枝、右左枝、右右枝，主干的分杈处称为“主节点”、左干的分杈处称为“左节点”、右干的分杈处称为“右节点”。

海龟的方向按顺时针依次称为：上、右上、右、右下、下、左下、左、左上。

4. “LCS 60”的执行过程

执行过程的叙述见表 7－4。

表 7－4 “LCS 60”的执行过程中各阶段信息一览表

递归深度				当前行执行的命令	执行当前命令后的绘图效果或执行结果	执行当前命令后海龟的方向
一层	二层	三层	四层			
L＝60 行号	L＝30 行号	L＝15 行号	L7.5 行号			
1				60＜10	不成立	上
2				FD 60 LT 45	画主干	左上
3				LCS 30	递归　进入二层	
	1			30＜10	不成立	
	2			FD 30 LT 45	画左干	左
	3			LCS 15	递归　进入三层	
		1		15＜10	不成立	
		2		FD 15 LT 45	画左左枝	左下
		3		LCS 7.5	递归　进入四层	
			1	7.5＜10	成立　退回三层	
		4		RT 90		左上
		5		LCS 7.5	递归　进入四层	
			1	7.5＜10	成立　退回三层	
		6		LT 45 BK 15	海龟退到左节点	左
		7		END	返回二层	
	4			RT 90		上
	5			LCS 15	递归　进入三层	
		1		15＜10	不成立	
		2		FD 15 LT 45	画左右枝	左上

续表 7－4 “LCS 60”的执行过程中各阶段信息一览表

递归深度				当前行执行的命令	执行当前命令后的绘图效果或执行结果	执行当前命令后海龟的方向
一层	二层	三层	四层			
L＝60 行号	L＝30 行号	L＝15 行号	L7.5 行号			
		3		LCS 7.5	递归　进入四层	
			1	7.5＜10	成立　退回三层	
		4		RT 90		右上
		5		LCS 7.5	递归　进入四层	
			1	7.5＜10	成立　退回三层	
		6		LT 45 BK 15	海龟退到左节点	上
		7		END	返回第二层	
	6			LT 45 BK 30	海龟退到主节点	左上
	7			END	返回第一层	
4				RT 90		右上
5				LCS 30	递归　进入二层	
	1			30＜10	不成立	
	2			FD 30 LT 45	画右干	上
	3			LCS 15	递归　进入三层	
		1		15＜10	不成立	
		2		FD 15 LT 45	画右左枝	左上
		3		LCS 7.5	递归　进入四层	
			1	7.5＜10	成立　退回三层	
		4		RT 90		右上

续表 7－4 “LCS 60”的执行过程中各阶段信息一览表

递归深度				当前行执行的命令	执行当前命令后的绘图效果或执行结果	执行当前命令后海龟的方向
一层	二层	三层	四层			
L＝60 行号	L＝30 行号	L＝15 行号	L7.5 行号			
		5		LCS 7.5	递归　进入四层	
			1	7.5＜10	成立　退回三层	
		6		LT 45 BK 15	海龟退到右节点	上
		7		END	返回第二层	
	4			RT 90		右
	5			LCS 15	递归　进入三层	
		1		15＜10	不成立	
		2		FD 15 LT 45	画右右枝	右上
		3		LCS 7.5	递归　进入四层	
			1	7.5＜10	成立　退回三层	
		4		RT 90		右下
		5		LCS 7.5	递归　进入四层	
			1	7.5＜10	成立　退回三层	
		6		LT 45 BK 15	海龟退到右节点	右
		7	END	返回第二层		
	6			LT 45 BK 30	海龟退到主节点	右上
	7			END	返回第一层	
6				LT 45 BK 60	海龟退到树根	上

从以上“走程序”的叙述可见，递归过程的执行中，海龟绕了一圈又回到了出发点。

采用表格分析法，各层递归调用层次清晰，不易出错，可作为分析递归程序的工具使用。

三、考考你

下面留给读者一个三层递归过程，供分析练习。

```
TO SHU1 L
  IF :L<5 STOP
  FD :L LT 70
  SHU1 :L/2
  RT 70
  SHU1 :L/2
  RT 70
  SHU1 :L/2
  LT 70 BK :L
END
```

执行“SHU1 60”将画出如图 7－15 所示的三权树。你能分析清楚它的运行经过吗？

图 7－15　三权树

第六节　直接递归和间接递归

递归除按上述的递归位置和次数分类外，还可以按递归方法分为直接递归和间接递归。

所谓“直接递归”，是指在递归过程的过程体中直接调用自己，相应的过程则称为“直接递归过程”。前面所介绍的递归过程都属直接递归过程。

如果一个过程在它的过程体中并不直接调用自身，而是通过调用一个或几个其它过程，最后调用到自身过程的递归称为“间接递归”，相应的过程则称为“间接递归过程”。

请看下面例题：

```
TO Q N
  HT
  IF :N=0 STOP
  X :N
END

TO X N
  FD 50
  REPEAT 3 [FD 30 RT 120]
  BK 20
  RT 60
  Q :N-1
END
```

执行“Q 6”的结果是海龟画出如图 7—16 所示的六面小旗。

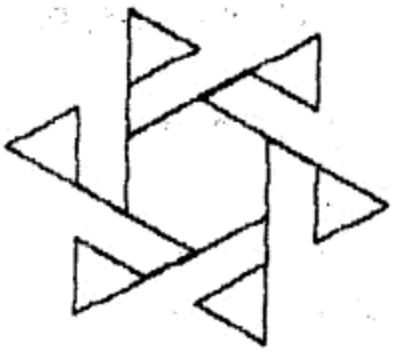


图 7—16　六面旗

在过程“X”中，调用了过程“Q”，而在过程“Q”中，又调用了过程“X”。也就是说，过程“X”通过调用过程“Q”实现了递归调用，所以“X”是间接递归过程。对等地看，过程“Q”也通过调用过程“X”而实现了递归调用，因此“Q”也是一个间接递归过程。请读者自己分析执行“X 6”的情况。

第七节　递归的深入讨论

在介绍了上述递归过程后，可能有些读者认为递归调用就是重复。其实不然，递归与重复决不是一回事。实际上递归是一种嵌套，执行递归过程，同样是按顺序执行过程的内容。下级过程调用上级过程时，必须在被调用的过程停止后（即执行到过程的 STOP 或 END 命令）才能继续执行，并且从调用之后的一条命令继续执行下去。各级递归调用都有自己独立的数据，决不会相互混淆。为了弄清递归的概念，我们分析一下例 7—10 过程的执行情况。

一、例 7－10 的中间递归过程分析

执行“L2 40”时，即令 B＝40，接着判断 B 是否小于 5，因为条件表达式的值为假，则相当于执行下面命令序列：

```
REPEAT 2 [FD 20 RT 90 FD 20]
RT 180
L2 20
REPEAT 2 [BK 20 RT 90 BK 20]
LT 180
HT
END
```

结果使海龟画出图 7－17 所示的图形(REPEAT 2 [FD 20 RT 90 FD 20])，并右转 90 度角；调用“L2 20”，返回后，再画出图 7－18 所示的图形(REPEAT 2 [BK 20 RT 90 BK 20])，并左转 90 度角；然后执行 END 命令结束。同样道理，调用“L2 20”相当于调用执行命令序列：

```
REPEAT 2 [FD 10 RT 90 FD 10]
RT 180
L2 10
REPEAT 2 [BK 10 RT 90 BK 10]
LT 180
HT
END
```

结果也是使海龟画出如图 7－19 所示的图形，继续调用“L2 10”，返回后再画出如图 7－20 所示的图形；最后执行 END 命令。

同样的方法依次类推，由于每递归一次使 B/2，一直递

归调用到执行“L2 2.5”(即 B=2.5)时,条件表达式 B<5 的值为真,则执行 STOP 命令,返回上一级递归过程 B=5 的状态,再从调用后的下一行命令继续执行。即执行:

```
REPEAT 2 [BK 5 RT 90 BK 5]
LT 180
HT
END
```

当执行到 END 时,过程“L2 5”执行完毕,返回到递归过程 B=10 时的状态。即执行:

```
REPEAT 2 [BK 10 RT 90 BK 10]
LT 180
HT
END
```

同理,执行 END 后,返回到过程 B=20 时的状态,即执行:

```
REPEAT 2 [BK 20 RT 90 BK 20]
LT 180
HT
END
```

最终使海龟画出图 7-10 所示的图形。

二、递归过程的深入探讨示例

下面我们再举几个例子,希望读者能在这些例题中得到一定的启发。由于篇幅所限,不能将所有的例题都详细解释,有的只作简单的说明,读者可以上机验证,仔细观察它的运行经过,最好能逐个过程作出总结,为以后自己编写程序打下基础。

图7－17

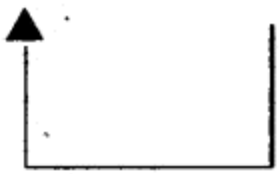


图7－18

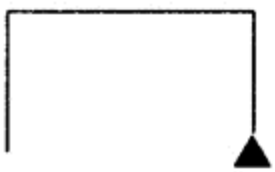


图7－19

图7－20

1. **宝塔**

例 7－12　编写画塔的过程。

```
TO TA :B :J
  IF :B<5 STOP
  REPEAT 3 [FD :B RT 90] LT 90
  BK :B+:J LT 45 FD 2 * :J RT 45 FD :B－:J RT
       180
  TA :B－:J RT 45 BK 2 * :J LT 45 FD :B+:J
  REPEAT 3 [BK :B RT 90]
  RT 90
  HT
```

```
END
```

如果执行：

```
? LT 90
? TA 20 4
```

将画出图 7－21 所示的宝塔的图形。请读者按照以上分析过程的方法，分析该过程的执行情况；然后命令计算机执行命令验证分析结果。

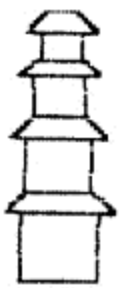


图 7－21

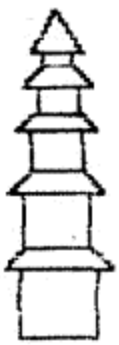


图 7－22

图 7－23

如果将过程“TA”作一简单的修改，将会画出一个新的塔形。具体的改动是在条件转移语句行的 STOP 命令之前，调用 TD 过程。改动的全部过程及“TD”过程如下：

```
TO TA :B :J
  IF :B<5 TD :B STOP
  REPEAT 3 [FD :B RT 90] LT 90
  BK :B+:J LT 45 FD 2 * :J RT 45 FD :B－:J RT
       180
  TA :B－:J RT 45 BK 2 * :J LT 45 FD :B+:J
  REPEAT 3 [BK :B RT 90]
```

```
    RT 90
    HT
END

TO TD :B
    REPEAT 3 [FD :B RT 90] LT 90
    REPEAT 3 [FD :B+3 LT 120 FD :B+3]
    REPEAT 3 [FD :B RT 90] LT 90
END
```

不难看出，这时再执行

```
? LT 90
? TA 20 4
```

就会画出图 7－22 所示的图形，与没改动前增加了图 7－23 的部分。实际上图 7－22 就是图 7—21 和图 7－23 两个图形拼在一起的综合效果。如果单独执行过程“TD 20”将会画出图 7－23 所示的图形。

2. **三杈树**

例 7－13　编写画三杈树的过程。

前面我们已经介绍过两杈树，并且作了过程分析。虽说是树，但仔细看起来总感觉不像。现在，我们来编写绘制比较逼真的三杈树的过程。

编程思路是：

用一个实心长方形画树杈，长方形的宽是长的六分之一，使得树枝越短越细，因此要单编一个过程“SZ”画实心长方形。

在主过程“SHU”中嵌套“SZ”过程，用三重递归画出三杈树。

开始先画树干，它与树枝的画法是一样的，越往上画树干越细。

设置参数 F，表示风力的大小。F 等于零时表示无风，树是直的；F 为正数时，向左转角减小，向右转角增大，树枝就向右偏，表示有风；F 为负数时，向右转角减小，向左转角增大，树枝就向左偏，表示风向与刚才的风向相反。F 的值越大，树的弯度就越大，表示风越大。

过程清单如下：

```
TO SZ B1 B2
  HT
  IF :B2<1 FD :B1 STOP
  REPEAT 2 [FD :B1 RT 90 FD :B2 RT 90]
  SZ :B1 :B2-1
END

TO SHU B1 F
  HT
  IF :B1<5 STOP
  SZ :B1 :B1/6
  LT 30-:F
  SHU :B1*0.6 :F
  RT 30
  SHU :B1*0.8 :F
  RT 30
  SHU :B1*0.6 :F
  LT 30+:F BK :B1
END
```

执行“SHU 30 0”可画出图 7－24 所示的静止树；

执行“SHU 30 6”可画出图 7－25 所示的、处于微风状态下的树；

执行“SHU 30 15”可画出图 7－26 所示的、向右倾斜的树(处于从左边刮来的、强风状态下)；

执行“SHU 30 ″－8”可画出图 7－27 所示的、向左倾斜的树(处于从右边刮来的、强风状态下)。

图 7－24　静止树

图 7－25　微风下的树

图 7－26　强风下的树 1

图 7－27　强风下的树 2

3. 相似三角大聚会

例 7－14　在一个等边三角形的每个角上，再递归画出较小的三角形，从而构成如图 7－28 所示的由三角形组成的递归图形。

过程如下：

```
TO SJ :N :B :F
  IF :N=0 STOP
  REPEAT 3 [FD :B RT 120 * :F SJ :N-1 :B/2 :F]
END
```

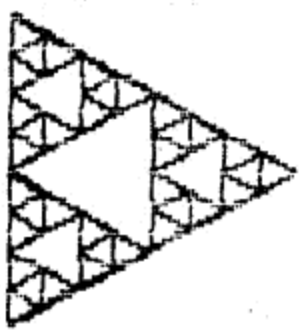


图 7－28　相似三角形

这里的 B 是边长参数，N 为级数参数，F 为符号参数。递归出现在重复命令中，当然也是多重递归。

4. 蒲公英

例 7－15　编写一个十次递归过程画一朵蒲公英。

过程如下：

```
TO Y R
  REPEAT 2 [REPEAT 9 [RT 5 FD 0.174 * :R RT 5]
RT 90]
END

TO HD N B
```

```
  IF :N=0 STOP
  REPEAT 10 [FD :B HD :N-1 :B*0.3 BK :B RT
  36]
END

TO YZ
  BK 50 Y 15 BK 20
  LT 90 Y 20 RT 90 BK 20
  RT 90 FD 40 BK 80
  HT
END

TO BY
  FD 30
  HD 2 20
  YZ
END
```

图 7—29　蒲公英

其中“HD”是一个十次递归过程，N 为级参数，B 为画线长度参数；过程“Y”是画一片叶子的过程；“YZ”是画花茎、叶子的过程；“BY”是主过程。

执行“BY”将画出如图 7－29 所示的图形。

5. **飞龙曲线**

例 7－16　编写绘制图 7－30 所示飞龙曲线的过程。

过程如下：

```
TO FL1 B N
  IF :N=0 FD :B STOP
HT FL1 :B :N-1
LT 90
FL2 :B :N-1
END

TO FL2 B N
  IF :N=0 FD :B STOP
  FL1 :B :N-1
  RT 90
  FL2 :B :N-1
END
```

这个程序写起来非常简单，但是执行起来很复杂。它有两个过程，这两个过程相差在一个是左转一个是右转，两个过程各自递归又互相调用。可以执行“FL1”也可以执行“FL2”。实际上这个过程就是我们前面介绍的间接递归。

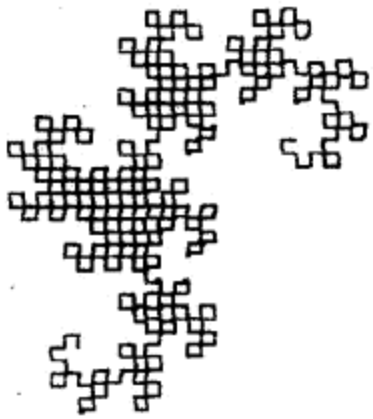


图 7－30　飞龙曲线

三、递归过程总结

通过上述的一些例子，我们可以总结出递归调用的一些规则：

第一、当一个过程被调用时，调用它的过程必须在被调用的子过程停止后，才能继续执行，并且在调用之后的下一条命令继续向下执行。

第二、在调用过程时，会出现一系列相同的过程临时共存的现象。

第三、这些相同过程具有自己独立的数值，不会相互混淆。

第四、递归调用时，当前有关的数据进入堆栈保存；当递归返回时，堆栈中的数据将被释放，使之返回到上一次调用它的地方。

第五、运用递归，可以使一些很复杂的问题简单化。

第八章　系统管理

当你编写了一个较复杂的过程时，当你指挥海龟画出了一个美丽的图画时，当你想把自己的作品与朋友交流时，当你想检查已经编写的全部过程时，……，可能会想到：如何将这些程序或图形保存起来呢？

本章将讨论解决这些问题的办法，将学习的新指令有：

单个过程清单列表指令	PO
所有过程清单列表指令	POALL
过程名列表指令	POTS
删除所有过程指令	ERALL
删除指定过程指令	ER
打印图形指令	OUTPRN
磁带存储指令	SAVE
磁盘存储指令	SAVE "A:
装入磁带文件指令	READ
装入磁盘文件指令	READ "A:

第一节　工作空间管理

“工作空间”是指电脑内存储器中，除去 LOGO 系统占用部分所剩下的区域，这部分区域是留给用户存放编制的程序、数据和图形的。简言之，它是供用户使用的内存空间。

实际操作中，用户常常需要了解工作空间的情况，如查找

已定义的过程，修改或删除某些过程等等。这就要使用 LOGO 的显示、打印、删除和编辑等指令。

一、过程列表指令

过程列表指令有三条，下面分别讨论。

1. 单个过程清单列表指令 PO

格式：PO 过程名

类型：命令

功用：打印指定过程的程序清单。

注释：PO 来源于“Print Order”，是“打印命令”的意思。在此表示显示过程。

说明：过程名必须是工作空间中已定义的过程。

示例：若用户已经定义了画五角星的过程，过程名为“WJX”。当键入：

```
? PO WJX          (回车后，将显示
TO WJX B
  HT
  REPEAT 5 [FD :B RT 144]
END
```

2. 所有过程清单列表指令 POALL

格式：POALL

类型：命令

功用：显示工作空间中所有过程的清单。

注释：POALL 来源于“Print Order ALL”，是“打印所有命令”的意思。

说明：POALL 命令能列出工作空间中所有过程的清单及过程名。当工作空间的内容很多时，将连续显示，直到工作

空间的内容全部显示完为止。

3. 过程名列表指令 POTS

格式:POTS

类型:命令

功用:显示工作空间中所有的过程名。

注释:POTS 来源于"Print Order Title Series ",是"打印命令名称序列"的意思。

说明:POTS 命令只能显示工作空间中已定义的所有过程名。

二、删除指令

删除指令的作用是清除工作空间中的内容。

1. 删除所有过程指令 ERALL

格式:ERALL

类型:命令

功用:删除工作空间中所有的过程及变量。

注释:ERALL 来源于"ERase ALL procedure",是"清除所有过程"的意思。

2. 删除指定过程指令 ER

格式:ER 过程名

类型:命令

功用:删除被指定的过程。

注释:ER 来源于"ERase",是"擦除、清除"的意思。

示例:假设工作空间中已存有过程"WJX",键入:

```
      ? ER WJX
```

回车后,"WJX"过程就被删除掉了。为了检查是否真的被删除掉,可用 POALL 命令或 POTS 命令查看。

删除操作时，一定要慎重，防止误删了有用的过程。最好把已经编好的程序或准备修改的一些过程先存入磁盘或磁带上，再整理工作空间。这样万一失误，也可以从磁带或磁盘上重新获得。

第二节　文件管理

内存中的过程不能长期保留，只要一关机，内存中的内容就会全部消失。要想长期保存，就要建立文件存储到外部存储器中，电脑学习机最常用的外存储器是磁带和磁盘。

所谓“文件”，即一些信息（程序或数据）的集合。它存放在外部介质（如磁盘、磁带）上。当需要存取某文件时，只要在一定状态下按规定格式指定文件名即可，而不必了解该文件的信息在外部介质上具体的物理位置。学会存取文件的操作方法，就可避免重复输入程序等复杂劳动，提高学习和工作效率。

一、程序的存储

“裕兴新世纪”的 LOGO 程序有两种存储方式：磁带和磁盘。

1. **磁带存储指令** SAVE

格式：SAVE ”文件名

类型：命令

功用：将工作空间内所有的过程作为一个文件存入磁带。

注释：SAVE 在英文中为“服务”的意思，在此表示“文件存储”。

说明：文件名是由首字符为大写英文字母组成的字符串，文件名允许与过程同名。在使用 SAVE 命令时，文件名的前面

一定要加双撇号。

示例：

? SAVE "WJM　　（文件名为 WJM

操作方法：

按 BASIC 程序存储方法连接录音机，键入 SAVE 和文件名之后，按下录音机的放音键 PLAY 和录音键 REC，待磁带走带平稳后按下回车键。这时，程序开始装入磁带。在程序装入磁带的过程中，机器会发出“嘟……”的叫声，当屏幕上显示 LOGO 提示符“?”和光标时表示装入完毕，按下录音机的停止键 STOP 键。

2. 磁盘存储指令 SAVE "A：

格式：SAVE "A：文件名

类型：命令

功用：将工作空间内所有的过程作为一个文件存入磁盘。

注释：SAVE 在英文中为“服务”的意思，在此表示“文件存储”。

SAVE 后的“"A：”为磁盘存储格式符号，文件名前不须加任何符号。

示例：

? SAVE "A：WJM　　（文件名为 WJM

操作方法：

将磁盘插入软盘驱动器（磁盘上的写保护口要关闭），键入上述命令按回车键后，软驱指示灯闪亮、磁盘开始转动，文件开始存入磁盘。由于磁盘干扰，存储过程中可能出现显示紊乱的情况，这是正常现象。存储结束，磁盘停止转动，磁盘指示灯熄灭，屏幕恢复正常显示，并显示 LOGO 提示符“?”和光标，表示文件存储完毕。

二、程序的调入

程序调入是指将存储在磁带或磁盘上的 LOGO 程序装入内存。

1. 装入磁带文件指令 READ

格式:READ "文件名

类型:命令

功用:将磁带上指定"文件名"的程序装入电脑工作空间。

注释:READ 在英文中为"读"的意思,在此表示从磁带读出文件。

说明:在使用 READ 命令时,在"文件名"前面一定要加双撇号。将新的 LOGO 程序装入工作空间后,原工作空间的过程就被清除掉。

示例:

? READ "WJM　　(WJM 为文件名)

操作方法:

录音机准备好后,键入 READ 和文件名,按回车键,同时按下录音机的放音键 PLAY,程序开始装入电脑内存。在此过程中会发出"嘟……"的叫声,装入完毕,屏幕上显示 LOGO 提示符"?",按下录音机的停止键 STOP。

如果读到的文件不是你所要的文件时,屏幕上会显示 SKIP,意思是让你跳过这段磁带,继续装入,直到装入 READ 语句所指定的文件为止;如果没有给出文件名,READ 命令只将从磁带读到的第一个文件调入内存。

2. 装入磁盘文件指令 READ "A:

格式:READ "A:文件名

类型:命令

功用：将磁盘上指定“文件名”的程序装入电脑工作空间。

注释：READ 在英文中为“读”的意思，在此表示从磁盘中读出文件。

READ 后的“"A：”为磁盘调入格式符号，文件名前不须加任何符号。

示例：

？READ "A：WJM　　（WJM 为文件名）

操作方法：

将存有文件名的磁盘插入软盘驱动器，键入示例命令回车后，软驱指示灯闪亮，磁盘开始转动，文件开始调入内存。当显示 LOGO 提示符“？”和光标后，文件调入完毕，软驱指示灯熄灭。现在使用过程列表命令即可列出过程清单。

第三节　图形及程序打印

这里的“打印”，是指通过打印机印出信息。打印图形就是绘制的图形通过打印机打印在纸上；打印程序是指将过程清单打印在纸上。

一、打印图形

打印图形指令 OUTPRN

格式：OUTPRN

类型：命令

功用：把屏幕上显示的图形打印在纸上（屏幕硬拷贝）。

注释：OUTPRN 来源于“OUT PRiNt”，是“输出到打印机”的意思。

打印机准备好后，将所要打印的图形显示在屏幕上，键入

OUTPRN 命令，打印机就会把图形打印出来。

二、打印程序

打印过程清单时，首先连接好打印机，执行打印机联机命令；然后根据程序打印类型，键入相应的列表命令。在打印机联机的情况下，显示命令就变成了打印命令。

连接打印机命令如下：

格式：Ctrl—P

类型：热键

功能：打印机联机。

说明：按一次组合键 Ctrl—P，使电脑系统与打印机连接，再按一次则断开系统连接。

示例：打印一个过程的清单的操作步骤如下：

- 按组合键：Ctrl—P　　（屏幕上并不显示任何字符）
- 键入：PO 过程名　　（工作空间必须有此过程）
- 回车，开始打印指定的过程清单。
- 等屏幕显示 LOGO 提示符时，打印完毕。

再如：打印工作空间的全部过程清单步骤如下：

- 按组合键：Ctrl—P
- 键入：POALL
- 回车后开始打印工作空间所有的过程清单。
- 等屏幕显示 LOGO 提示符时，打印完毕。

又如：打印工作空间过程名清单的步骤如下：

- 键入组合键：Ctrl—P
- 键入：POTS
- 回车后开始打印工作空间所有的过程名。
- 等屏幕显示 LOGO 提示符时打印完毕。

第九章　数值计算

LOGO 语言不仅具有很强的绘图功能，而且还具有很强的计算功能。本章将讨论数学运算程序的编写方法和技巧。

结合数值运算的讨论，本章将介绍如下指令：

换行显示指令	PR
紧凑显示指令	PRINT1
显示定位指令	CURSOR
结果返还指令	OP
输出变量值函数指令	THING
变量赋值指令	MAKE
平方根函数指令	SQRT
取整函数指令	INTEGER
四舍五入取整函数指令	ROUND
先四舍五入后求商取整函数指令	QUOTIENT
先四舍五入后取余函数指令	REMAINDER
求字符的 ASCII 码函数指令	ASCII
求 ASCII 码对应的字符函数指令	CHAR

第一节　LOGO 中的数及变量

LOGO 中的数有整数、实数两种。

一、整数

不带小数点的数称为整数，如：

5，13，－123

等都是整数。由于机器字长和系统软件的差别，不同机型 LOGO 语言中的整数范围也不同。“裕兴”LOGO 系统的整数取值范围是－2147483647～＋2147483647。虽然 LOGO 已经规定了整数范围，还可以将超出整数范围的数据输入电脑，只不过这时的数据将以实数的形式存放和显示。

二、实数

LOGO 中的实数即是数学中的小数。如

8.12，0.73，－20.37，5.0

等都是实数。实数的范围是$\pm 10^{-18}$～$\pm 10^{18}$，如果超出这个范围，就会发生溢出错误。LOGO 的运算结果只能输出 10 位(包括小数)，如果超出这个界限，将采用科学记数法表示，请看下列数的表示：

897600	可表示为	8.976E6
－0.0008927	可表示为	－8.927E－03
314.159	可表示为	3.14159E2

上面的 E6 表示乘以 106，E－03 表示乘以10^{-3}。“E ”表示以 10 为底的指数。用这种形式表示实数的一般形式为：

±整数部分.小数部分 E 指数

整数和实数的区别，主要在于表示方式，与数学中整数和实数的概念有所不同。整数的加、减、乘运算的结果仍是整数(只要结果不超过允许的范围)。这时的结果是精确值，有效数

字可达十位。但整数相除总产生实数，即使所得商是整数也以实数方式表示。

三、变量

变量是在程序执行过程中取不同数值的量。LOGO 语言中的变量是用“:”后紧跟变量名表示的，整个变量的字符串中不允许有空格。变量名由除方括号、圆括号、运算符以外的字符组成。如：

:A　　:AB　　:BLM3

等都可以作为变量，通常形式是字母数字串。

定义一个变量，将在内存中分配给连续的部分存储单元，存储单元的多少取决于给变量赋值的类型，若给变量赋以数值，将给该变量分配 4 个单元；若赋给字符（以双撇号开始）则分配 256 个单元。也就是说，LOGO 中的变量既可以存放数值也可以存放字符。

变量中的内容取之不尽、用之不完。也就是说，当从变量中读出内容时，变量的内容仍保持完好。但当有新的内容送到这个变量时，旧的内容将被冲掉，而为新的内容所代替。关机后变量中的内容将被清除。

第二节　算术运算和表达式

运算分为算术运算、关系运算和逻辑运算三种。关系运算和逻辑运算我们在第七章里已经叙述过，在这里只介绍算术运算和算术表达式。

一、算术运算符

LOGO 语言中有加(+)、减(-)、乘(*)、除(/)四种运算。它的表达形式和规则见表 9-1。

表 9-1 LOGO 中的运算符

运算符	运算内容	示例	数学表达式
+	加	:A+:B	A+B
-	减	:A-:B	A-B
*	乘	:A*:B	A×B
/	除	:A/:B	A÷B

加法、减法的表达形式与数学上相同。乘法符号不能写成“×”或“·”,也不可省略乘号不写。除号不能写成“÷”。在LOGO 语言中没有乘方运算,不适于进行一些很复杂的计算,在算整数幂时,可用乘法代替,如 $2^3=2*2*2$。

运算的优先顺序为:

(1)( )

(2)“*”、“/”

(3)“+”、“-”

乘法和除法是同一级运算;加法和减法是同一级运算。在同一个算式,如果出现同级的运算时,应按从左至右的顺序进行运算。要改变运算次序,可用加圆括号的办法来解决。有圆括号的算式,应先圆括号内,后括号外。圆括号可有多层,运算时应由最里面的圆括号开始,逐层向外运算。表达式中不能用方括号和花括号。

二、算术表达式

由常数、变量、函数(在后面介绍)和算术运算符组成的各种式子就称为算术表达式。表达式的写法与数学中有区别,请见表 9－2。

表 9－2　LOGO 中的算术表达式

数学表达式	LOGO 表达式
X+Y	:X+:Y
3a+2b−1	3 * :A+2 * :B−1
$1-\frac{x}{4}+\frac{x*x}{5}$	1−:X/4+:X * :X/5
$\frac{A-B}{A+B}$	(:A−:B)/(:A+:B)

上表所列出的都是一些基本表达式,在进行数学计算时,可以采用直接命令或编写程序两种方式。

如:要计算出一个表达式的值,可在命令状态下直接输入:

? 4+6 * 13/6.5−17

回车后,会立即显示出

RESULT: −1

RESULT 是"结果"的意思。

这种执行方式,输入的可以是一个表达式,但表达式中不能含有变量,只能是常数和标准函数。

第三节　输　　出

使用电脑进行计算时，总要将计算结果通过输出设备表示出来。LOGO 语言的输出命令是 PR，它能将运算结果显示在屏幕上，并将字符串原样显示出来。

显示语句的命令有两个：PR 和 PRINT1。下面分别介绍。

1. **换行显示指令** PR

格式：PR＜空格＞＜输出内容＞

PR＜空格＞[＜字符串＞]

类型：语句、命令

功用：显示运算结果或字符串。

注释：PR 来源于“PRint”，是打印的意思，在此表示把结果打印在屏幕上。

PR 命令与输出内容之间必须有一个空格，这是语句格式规定的，不能省略。

输出内容可以是字符串，也可以是算术表达式。当输出内容是字符串时，字符串的前面要加双撇号“””（第一种格式），也可以使用方括号“[ ]”括起来（第二种格式）；当输出内容是算术表达式时，PR 命令将首先计算出表达式的值，然后再输出到屏幕上。

示例：

? PR 5 * 2 ↙　　（屏幕显示

10

? ■

当键入：

? PR ″2 * 5　　（将显示

2 * 5

? ■

可见，输出内容前面有双撇号时，系统已把算术表达式作为字符串看待。

使用方括号同样把其中的内容作为字符串输出。

当键入：

? PR [THIS IS A PEN]　　(显示

THIS IS A PEN

? ■

注意：采用第一种格式时，只在显示的内容前有一个双撇号"，后面没有，且在显示的字符中不能有空格。采用后一种格式时，只要将要显示的内容括起来，这时，不论有无空格，方括号内的字符都可全部显示出来。

2. **紧凑显示指令** PRINT1

格式：PRINT1＜空格＞＜输出内容＞

类型：语句、命令

功用：以紧凑格式显示输出内容，显示完后不换行。

说明：所能显示的内容与PR 命令相同。只是PR 在显示完成时光标移到下一行(换行)，而 PRINT1 在显示完时光标停留在当前显示内容的后边，下一次显示时，接着向下显示。

示例：

? PRINT1 (15+7)/2　　(回车后，将显示：

11

? ■

要想显示一个完整的算式(15+7)/2=11，可键入：

? PRINT1 "(15+7)/2= PR (15+7)/2

回车后，将显示：

? (15+7)/2=11

? ■

3. **显示定位指令** CURSOR

格式:CURSOR　参数 1　参数 2

类型:语句、命令

功用:把光标移到以“参数 1”的值为列,“参数 2”的值为行的位置上。

注释:CURSOR 在英文中为“光标”的意思。

说明:参数 1 定位横坐标,取值范围 2～29,参数 2 定位纵坐标,取值范围为 3～27。两个参数均可以取数值常数、变量或表达式。

示例:键入

```
? CURSOR 15  14
```

回车后,光标将移到屏幕的中央。

使用光标定位指令,可以编写一个显示图 9－1 所示图形的过程。过程清单如下:

```
TO SJX X Y Z
  IF :X>10 STOP
  IF :X<2 STOP
  CURSOR :X :Y
  REPEAT :Z [PRINT1 :X-1]
  PR ”
  SJX :X-1 :Y+1 :Z+2
END
```

其中,参数 X 控制每一行开始显示的列位置,参数 Y 控制显示的行位置,每执行一次过程体向下移一行,参数 Z 控制每次显示数字的个数。

执行“SJX 10 3 1 ”即可打出图 9－1 所示的数字金字塔。

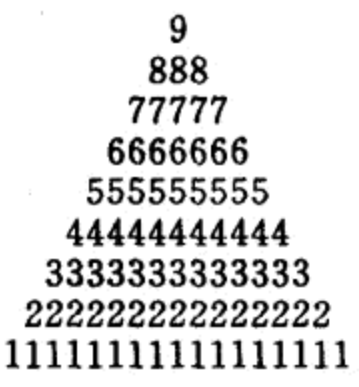


图 9－1　数字金字塔

这个程序可以显示从 1 到 9 开始的不同层数的数字三角形，如执行“SJX 5 3 1 ”就可显示出以 5 开始、下面各行依次为 4、3、2、1 的数字三角形。

4. **结果返还指令** OP

格式：OP 表达式

类型：语句

功用：终止当前过程的执行，把表达式的结果作为当前过程的信息，返还给调用它的过程。

注释：OP 来源于“OUTPUT”，是“输出放置”的意思。在此表示“把当前过程输出的内容，放置到调用过程中去”。

为了说明 OP 的功能，请看下面的示例。

例 9－1　我们先定义两个过程“CF”、“J1”如下：

```
TO CF X
  PR :X * :X
END

TO JI X Y
  PR :X+:Y
```

```
END
```

过程定义完毕，输入：

```
? CF1 4                    (回车后，将显示
16
```

再输入：

```
? JI 4 6                   (回车后，将显示
10
```

分别执行这两个过程，得到了各自的结果。但不能让“JI”过程的结果作为“CF”的输入量。如执行：

```
? CF JI 4 6
```

电脑将显示出错信息，告诉你“JI”的结果不能作为“CF”的输入量。

要实现过程之间的数据传递，就要使用“OP”命令。下面我们把上面的两个过程改写一下，用 OP 命令代替 PR 命令，观察它的结果。

```
TO CF X
  OP :X * :X
END

TO JI :X :Y
  OP :X+:Y
END
```

现在执行这个两个过程，键入：

```
? PR JI 10 20              (回车后，将显示
30
```

这里过程“JI”的计算结果通过 OP 输出传递到 PR 语句。

带 OP 输出命令的过程还可以作为其它过程的输入，如键

入：

? PR CF (JI 4 6)　　(回车后，将显示

100

执行上述命令时，首先执行过程“JI”计算两个参数的和；然后执行过程“CF”计算平方幂；最后由 PR 命令输出结果。从而完成了复合运算，实现了过程之间的数据传递。这些若不使用带结果返还指令“PO”的过程，是无法做到的。

在 LOGO 语言中没有乘方运算命令，我们可以用 OP 命令编写一个计算任意数的正整数次幂的运算过程。过程如下：

```
TO XN X N Y
  IF :N=0 OP :Y
  OP XN :X :N-1 :Y * :X
END
```

键入：

? PR XN 2 3 1　　(回车后，将显示

8

在这个过程中，变量 Y 是累乘器。过程执行后，Y 的值从 1 开始，每执行一次过程体，Y 乘上一个 X，执行 N 次后 Y 中的值就是 X×N。这是一个递归过程，与过程嵌套一样，要完成各次递归之间的数据传送，必须在递归时加上输出语句。条件转移语句中 OP 的作用是，在条件(N=0)成立时，将变量 Y 中的结果返还给调用本过程的命令或过程，它除了完成输出数据的工作，还代替了 STOP 命令控制递归结束。

5. **输出变量值函数指令** THING

格式：THING 参数

类型：函数

功用：输出参数的值。

注释：THING 在英文中为“东西、物品”的意思，在此表示“输出参数的内容”。

说明：参数必须是变量名。一般情况下变量前使用双撇号，不用冒号。

示例：

```
TO TH A
  PR THING "A
END
```

键入：

```
? TH 50                    (回车后，显示
50
```

注意：这里的“THING "A ”相当于“:A”。

第四节　赋　　值

1. 变量赋值指令 MAKE

现在介绍在过程体中给变量赋值的方法。

格式：MAKE "变量名 表达式

类型：语句、命令

功用：把表达式的内容赋给变量。

注释：MAKE 在英文中为“使、致使”的意思，在此表示“使变量等于表达式的值”。

双撇号用来指示赋值对象；双撇号后面是变量名而不是变量，不许带冒号；表达式是赋给变量的内容。

执行赋值语句时，首先计算表达式的值，然后把这个值赋给变量名所表示的变量。赋值内容还可以是字或表（字、表将在下一章中介绍）。

示例：

? MAKE "A 6 * 3/2

回车后，变量 A 的值等于 9。

说明：在赋值语句中，带双撇号的变量与其它情况下带冒号同名变量的含义是一样。之所以有双撇号和冒号的差别，缘于 LOGO 语言在赋值语句的变量前面必须使用双撇号的约定。

赋值表达式中可以有变量，但这个变量必须是先赋值的。

例如：

```
TO A
  MALE " X 5
  MAKE " Y :X+2
  PR :Y
END
```

执行过程 A，即可显示变量 Y 中的内容为：7。

又如：

```
TO B X1
  MAKE "X :X1
  PR :X
  B :X1+1
END
```

执行过程"B"是给参数 X1 赋以初值，将显示从 X1 开始的自然数序列(每递归一次，输出的结果加 1)。

2. 赋值语句的应用

MAKE 是一个非常有用的语句，请看下面几例。

例 9—2　求 N 项和

计算 1+2+3+…+N 的结果(求自然数序列的前 N 项

和)。

过程清单如"IN":

```
TO IN N I S
  IF :I>:N PR :S STOP
  MAKE "S :S+:I
  IN :N :I+1 :S
END
```

三个参数 N、I、S 分别代表累加数的项数、累加数、累加和。

键入:

```
? IN 100 1 0
```

将显示自然数前 100 项的累加和为 5050。

例 9—3　阶乘

计算任何自然数 N 的阶乘。

过程清单如"JC":

```
TO JC N Y
  IF :N=0 OP :Y
  MAKE "Y :Y * :N
  OP JC :N-1 :Y
END
```

"JC"的运算过程为降阶求积,变量 Y 存乘积,初值为 1,每一次递归 Y 乘一次 N、N 减 1,当 N=0 时结束递归,输出阶乘值。

键入:

```
? JC 5 1                    (回车后,将显示
RESULT 120
```

运行结果表明,5 的阶乘等于 120。

例 9－4　建造金字塔

取若干块方石，建造如图 9－2 所示的金字塔。

过程清单如下：

```
TO S Z X
  MAKE "Z :Z－:X
  IF :Z<0 OP :X－1
  OP S :Z :X+1
END

TO F X
  REPEAT :X [REPEAT 4 [FD 15 RT 90] RT 90
    FD 15 LT 90]
  RT 90 BK :X * 15－7.5 LT 90 PU
  FD 15 PD
END

TO L X
  IF :X<1 STOP
  F :X
  L :X－1
END

TO JZ Z
  HT IF :Z>100 STOP
  PU SETXY －100 0－70 PD
  MAKE "X S :Z 1
  L :X
```

END

“JZ”是主过程，变量 Z 是使用的石块数，当 Z＞100 时方石太多，画出的图形将超出屏幕，因此取 Z＞100 控制结束绘图；过程“S”的功能是计算 Z 块方石可建造多少层金字塔，输出层数赋给主过程中的变量 X；过程“F”控制画一层；过程“L ”控制画出整座金字塔。

执行“JZ 30”将画出图 9－2 所示的金字塔。

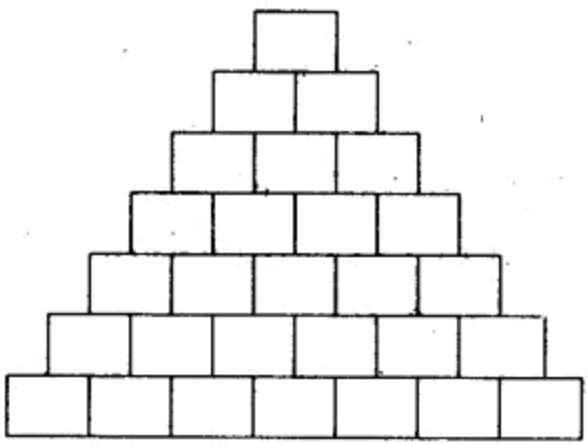


图9－2　金字塔

例 9－5　电子钟

编写一个具有指针和数字两种显示的电子钟程序。过程清单如下：

```
TO BP
  HT PU SETXY －85 0－70 PD
  REPEAT 4 [FD 170 RT 90]FD 85
  REPEAT 36 [RT 5 FD 85 * 6.28/36 RT 5]
  PU RT 90 FD 85 PD LT 90
```

```
    REPEAT 12 [PU FD 65 PD FD 15 PU BK 80 RT 30
PD]
  END

  TO Z X Y Z
    IF :Y>59 SETH :X * 30 PICKPEN 0 FD 30 BK 30
       MAKE "Y 0 MAKE "X :X+1
    IF :Z>59 SETH :Y * 6 PICKPEN 0 FD 45 BK 45
       MAKE "Y 0 MAKE "X :X+1
    PICKPEN 1 SETH :X * 30+:Y/2 FD 30 BK 30
    SETH :Y * 6 FD 45 BK 45
    SETH :Z * 6 FD 60 PICKPEN 0 BK 60 PICKPEN 1
    CURSOR 13 24 PRINT1 :X PRINT1 ": PRINT1 :Y
       PRINT1 ": PR :Z
    Z :X :Y :Z+1
  END

  TO BIAO X Y Z
    HT BP Z :X :Y :Z
  END
```

执行命令“BIAO 3 5 53”，将画出图9—3所示的电子钟。

执行过程“BIAO”，先调用“BP”过程画出表盘；再调用“Z”过程画出走动的表针，并显示出时、分、秒的数字，用擦了又画、画了又擦的方法表示表针的走动；“Z ”过程递归一次走一秒，要想控制走时的准确度，可改变“Z”过程的运行方式。

这个程序没有停止命令，程序一经运行，电子钟就一直走下去，只有按 BREAK 键才能打断程序的运行。

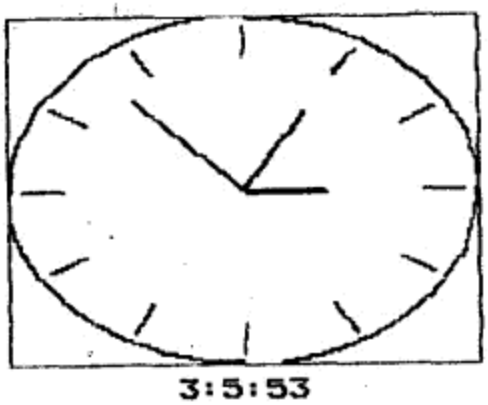


图9—3　电子钟

第五节　标准函数

LOGO 已把数学中常用的标准函数编成微程序放在系统中，用户进行函数运算时，可以直接调用。

一、平方根函数指令 SQRT

格式：SQRT 参数

类型：函数、命令、语句

功用：计算参数的算术平方根。

注释：SQRT 来源于“SQuare RooT”，是“平方根”的意思。

参数可以是数值常数、变量或表达式。但参数的值必须大于等于零。

示例：键入

?PR SQRT 9　　（回车后，将显示

3

表示9的平方根等于3。

例9－6　已知两圆的半径，画出如图9—4所示的两个圆（两圆互相外切、又都与一直线相切）。

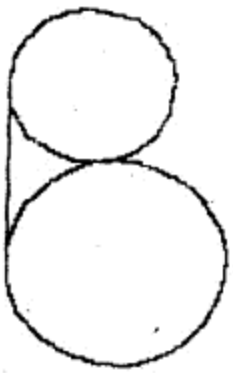


图9—4　相切圆

编程的关键是求出两圆和直线的两个切点之间的距离。

设两圆的半径分别为R1和R2，编写过程如下：

```
TO Y R
  REPEAT 36 [RT 5 FD :R * 6.28/36 RT 5]
END

TO XQ R1 R2
  Y :R1
  MAKE "X (:R1+:R2) * (:R1+:R2)
  MAKE "Y (:R1-:R2) * (:R1-:R2)
  FD SQRT (:X-:Y)
  Y :R2
END
```

执行“XQ 50 30”即可画出图9—4所示的图形。

二、取整函数指令 INTEGER

格式:INTEGER 参数

类型:函数、命令、语句

功用:对参数的值去尾取整。

注释:INTEGER 在英文中为"整数"的意思。

参数可以取数值常数、变量或表达式,执行时首先计算表达式的值,再将小数部分去掉,保留整数部分(INTEGER 函数不遵循四舍五入的原则)。

示例:键入

```
?PR INTEGER 3.95      (回车后,将显示
3
```

键入

```
?PR INTEGER 3.05      (回车后,将显示
3
```

运行结果表明了 INTEGER 函数的取整特点。

例9—7　判断一个数能否被另一个数整除。

两数 A、B,若有

$$A/B=INTEGER(A/B)$$

则说明 A 可以被 B 整除。过程如下:

```
TO ZC A B
  IF :A/:B = INTEGER :A/:B THEN PR "YES
ELSE PR "NO
END
```

执行"ZC 15 3",将显示"YES",表示15可以被3整除。

执行"ZC 15 2",将显示"NO",表示15不能被2整除。

三、四舍五入取整函数指令 ROUND

格式:ROUND 参数

类型:函数、命令、语句

功用:对参数的值进行四舍五入。

注释:数学中,ROUND 的英文含义是"约略数目字"的意思。在此表示四舍五入。

参数可以取数值常数、变量或表达式。执行时,首先计算表达式的值,然后将小数部分进行四舍五入。

示例:键入

?PR ROUND 3.5　　(回车后,将显示

4

若键入

?PR ROUND 3.49　　(回车后,将显示

3

若键入

?PR ROUND −3.5　　(回车后,将显示

−3

若键入

?PR ROUND −3.51　　(回车后,将显示

−4

四、先四舍五入后求商取整函数指令 QUOTIENT

格式:QUOTIENT 参数1 参数2

类型:函数、命令、语句

功用:先对两个参数四舍五入取整,再求商(参数1除以参数2),并对商进行去尾取整处理。

注释:QUOTIENT 在英文中为"商"的意思。

QUOTIENT 相当于 INTEGER（ROUND :X / ROUND :Y)。参数1、2可以取数值常数、变量或表达式。

注意:当参数2是负数时要加圆括号。

示例:

?PR QUOTIENT 8.5 3 （回车后,显示

3

键入

PR QUOTIENT 8.45 3 （回车后,显示

2

键入

PR QUOTIENT 6.5（－3回车后,显示

－3

键入

PR QUOTIENT 6.45（－3回车后,显示

－2

例9－8　求三个正整数的最小公倍数。

求 A、B、C 三数最小公倍数的方法是:任意取其中一个数,比如取 C,先看 C 能否被 A、B 整除,如果都能整除,C 就是三个数的最小公倍数;如果不能,再看2C 能否被 A、B 整除,如果都能整除,2C 就是最小公倍数;如果不能,再看3C……;一直进行下去,直到碰见第一个能被 A、B 整除的 nC 就是最小公倍数。

过程清单如下:

```
TO ZC A B C S
  MAKE "X QUOTIENT :S :A
  MAKE "Y QUOTIENT :S :B
```

```
    IF :S/:A=:X IF :S/:B=:Y PR :S STOP
    ZC :A :B :C :S+:C
END
```

执行“ZC 6 9 14 14”，回车后将显示“126”，说明6、9、14的最小公倍数为126。

五、先四舍五入后取余数函数指令 REMAINDER

格式：REMAINDER 参数1 参数2

类型：函数、命令、语句

功用：首先对两个参数进行四舍五入处理，然后求出参数1除以参数2的余数。

注释：在数学中，REMAINDER 的英文含义是“余数”的意思。

该函数将给出(ROUND :X)/(ROUND :Y)商的余数部分。其规则同求商取整函数相同。例如：键入

```
?PR REMAINDER 24 9          (回车后，将显示
6
```

键入

```
PR REMAINDER 25.5 13.2      (回车后，将显示
0
```

例9－9　韩信点兵

有一队士兵，不知人数。他们3人一排余2人，5人一排余4人，6人一排余5人，7人一排正好不余。问：这队士兵最少有多少人？

编程思路：

拿被7整除的数(从最小数开始)，去验证其它三个条件，第一次满足三个条件的数就是所求的结果。

过程清单如下：

```
TO DB X
  MAKE "A REMAINDER :X 3
  MAKE "B REMAINDER :X 5
  MAKE "C REMAINDER :X 6
  IF ALLOF (:A=2) (:B=4) THEN BB :X
  DB :X+7
END

TO BB X
  IF :C=5 THEN PR :X TOPLEVEL
  DB :X+7
END
```

过程定义完毕，键入“DB 7”，回车后，将显示

119

读者可以验证一下，119是否能同时满足以上四个条件。

六、正弦函数指令 SIN

格式：SIN 参数

类型：函数、命令、语句

功用：计算参数的正弦值。

注释：SIN 来源于“SINe”，是“正弦运算”的意思。

参数的单位为度，可以取数值常数、变量或表达式。取值为LOGO 所能处理的数据范围。

示例：键入

```
?PR SIN 90                    (回车后，将显示
1
```

结果说明，当参数取90度时，对应的正弦函数值为1。

七、余弦函数 COS

格式：COS 参数

类型：函数、命令、语句

功用：计算参数的余弦值。

注释：COS 来源于“COSine”，是“余弦运算”的意思。

参数的单位为度，可以取数值常数、变量或表达式。取值范围同 SIN。

示例：键入

?PR COS 60　　（回车后，将显示

.5（0.5）

结果表明，60°的余弦值是0.5。

八、反正切函数指令 ATAN

格式：ATAN 参数1 参数2

类型：函数、命令、语句

功用：计算反正切值。

注释：ATAN 来源于“Arc TANgent”，是“反正切运算”的意思。

参数1、参数2分别代表所求角度的对边及邻边的长度，可以取数值常数、变量或表达式，取值为 LOGO 所能处理的数据范围。ATAN 输出的是角度值，单位为“度”。

示例：键入

?PR ATAN 5 3　　（回车后，将显示

59.0368

输出结果表明，对边是5、邻边是3的直角三角形的角度是

59.0368°。

九、随机函数指令 RANDOM

格式:RANDOM 参数

类型:函数

功用:产生0至所给参数之间的随机整数。

注释:RANDOM 在英文中为"随机"的意思。

参数必须是正整数,可以取数值常数、变量或表达式的形式,取值为 LOGO 所能处理的整数范围。

示例:键入

```
?RANDOM 10
```

将显示出0到9中的某一个整数。

要想产生其它范围的随机数,要增加一些处理技巧。例如,要产生0到1之间的两位随机小数,可用下列命令:

```
?PR (RANDOM 100)/100
```

执行这条命令后,电脑先随机产生一个小于100的整数,然后再将它变成两位小数。

例9－10　编写产生10个50至100之间随机数的过程,并将其中的奇数打印出来。过程如下:

```
TO JI N
  IF :N>10 STOP
  MAKE "A 50+RANDOM 51
  IF REMAINDER :A 2=0 ELSE PR :A
  JI :N+1
END
```

执行"JI 1",将打印

```
51
```

```
75
75
81
55
```

注意：在使用随机函数时，表达式的值不能是负数，否则将会出现错误。

十、自定义函数

除以上介绍的系统标准函数外，我们可以用定义过程的方法自定义一些常用函数供使用。例如，LOGO 的标准函数中没有正切函数，我们可以用正弦函数和余弦函数来定义自己的函数(通常称为自定义函数)。

定义一个正切函数“TAN”，过程如下：

```
TO TAN X
  OP (SIN :X)/(COS :X)
END
```

退出定义后，我们就有了正切函数。现在键入

```
?PR TAN 0                    (回车后，将显示
0
```

结果表明，0°的正切值等于0。

显然，编程中如果需用正切函数，就可以直接调用“TAN”。

十一、求字符的 ASCII 码函数指令 ASCII

电脑中，每一个字符都对应着一个机器内码，通常称为 ASCII 码。当指定一个字符时，使用 ASCII 指令可以求出该字符对应的机器内码。

格式:ASCII "一个字符

类型:函数、命令、语句

功用:输出一个字符的 ASCII 码。

注释:ASCII 来源于" American Standard Code for Infomation Interchange",是"美国标准信息交换代码"的意思。

说明:ASCII 后面只能跟一个字符或变量(变量的值也只能是一个字符),如果是多个字符,将只显示第一个字符的 ASCII 码。

示例:键入

```
?PR ASCII "A                    (回车后,将显示
65
```

键入

```
?MAKE "A "ABC
?PR ASCII :A
65
```

运行结果表示字符"A"的 ASCII 码为65。

十二、求 ASCII 码对应字符函数指令 CHAR

格式:CHAR 参数

类型:函数、命令、语句

功用:输出该 ASCII 码对应的字符。

注释:CHAR 来源于"CHARacter",是"字符"的意思。

参数可以是数值常数、变量或表达式,它以256为模,因此"CHAR 65"与"CHAR 321"的结果相同。

示例:键入

```
?PR CHAR 65                     (回车后,将显示
A
```

运行结果表明 ASCII 码65对应的字符是“A”。

第六节　应用题举例

一、求解一元二次方程

例9－11　编写过程求一元二次方程

$$AX^2+BX+C=0$$

的解。

大家知道，一元二次方程式的实数解公式为

$$X1,2=(-B\pm\sqrt{B^2-4AC})/2A$$

求解时，首先要判别 B^2-4AC 的值是否小于0，如果小于0，则无实数根。过程如下：

```
TO JIE A B C
  MAKE "D (:B * :B－4 * :A * :C)
  IF :D<0 PR [mei you jie] STOP
  MAKE "X1 (－:B+SQRT :D)/(2 * :A)
  MAKE "X2 (－:B－SQRT :D)/(2 * :A)
  PRINT1 [X1=] PR :X1
  PRINT1 [X2=] PR :X2
END
```

如果 A，B，C 三个系数的值分别为“1、－1、－6”，则键入：

```
?JIE 1 (－1) (－6)                    (回车后，显示
X1=3
X2=－2
```

二、求三角形面积

例9－12　编写过程求三角形的面积。

```
TO SMJ A B C
  MAKE "P (:A+:B+:C)/2
  PR SQRT :P * (:P－:A) * (:P－:B) * (:P－:C)
END
```

过程中“A、B、C”分别为三角形的三条边长。

假定三角形的三边长依次为“3、4、5”，键入“SMJ3 4 5”，回车后，将打印该三角形的面积为“6”。

三、掷硬币

例9－13　投掷硬币。

这是数理统计中一个著名的试验。将一枚硬币投掷100次，出现国徽的次数和出现字样的次数各接近50次，掷的次数越多就越接近所掷次数的一半。这说明每次投掷中，出现国徽、字样的可能性各占二分之一。该试验也可以用电脑模拟，并统计出现的次数。

编程思路：

用“RANDOM 2”产生“0、1”之中的一个随机数，用“0”表示国徽面，用“1”表示字样面，然后统计各自出现的次数。过程清单如下：

```
TO TYB N M I Z
  MAKE "X RANDOM 2
  IF :X<0.5 THEN MAKE "N :N+1 ELSE MAKE
"M :M+1
  IF :I=:Z PRINT1 [gh:] PR :N PRINT1 [zi:]
```

```
    PR :M STOP
  TYB :N :M :I+1 :Z
END
```

过程中的变量“N”用来统计“0”出现的次数，“M”用来统计“1”出现的次数，“I”用来统计掷的次数，“Z”表示规定的投掷次数。执行

```
?TYB 0 0 1 100          (回车后，显示
gh: 56                  (国徽面
zi: 44                  (字样面
```

重复执行这个程序，每次显示的结果是不一样的，但将都向50靠近。

四、五针松盆景

例9－14　编写一个画松树盆景的过程。

```
TO CF B1 B2
  REPEAT 2 [FD :B1 RT 90 FD :B2 RT 90]
END

TO HP
  HT CF 4 115 RT 90 FD 5 RT 80
  FD 40 LT 80 FD 10 CF 10 4 FD 60
  CF 10 4 FD 20 LT 80 FD 40 LT 100 FD 60
END

TO S N
  IF :N<1 THEN STOP
  MAKE "X 65－RANDOM 120
```

```
  MAKE "Y 100-RANDOM 115
  PU SETXY :X :Y PD LT 60
  REPEAT 5 [FD 15 BK 15 RT 30] LT 90
  S :N-1
END

TO PJ
  PU SETXY -57 0-60 PD HP RT 95 PU
  FD 5 PD CF 30 2 FD 30 RT 15
  CF 30 2 FD 30 RT 40 CF 30 1
  LT 90 CF 50 1
  SETH 0 S 70
END
```

执行“PJ”，将画出如图9－5所示的盆景。

图9—5　松树盆景

过程说明：

“CF”过程用于画长方形；“HP”过程调用“CF”过程画花盆和树干；“S”过程用来画树叶。树冠的大小范围由随机函数决定(X 在－65到＋65之间、Y 在－10到＋100之间)。变量“N”控制松针数量。“PJ”是主过程，它调用“HP”过程画花盆，调用

“CF”过程画树干，调用“S”过程画松针。

五、电脑七巧板

例9－15　七巧板拼图。

1. 七巧板的结构

有一副七巧板(如图9—6所示)，有大三角形2个、中三角形1个、小三角形2个、平行四边形1个、正方形1个。使用这七块板可以拼凑各种各样的有趣图形。下面用电脑编写程序模拟七巧板拼图。

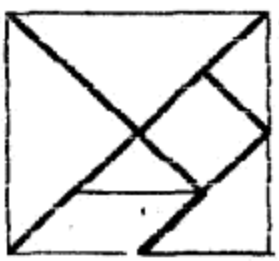


图9－6　七巧板

2. 各板边长设定

为方便编程，特对各块图板的边长设定如下：

大三角形的斜边长为 B，直角边长是 $B/\sqrt{2}$ ；

中三角形的斜边长为 $B/\sqrt{2}$ ，直角边长为 $B/\sqrt{2}$ ；

小三角形的斜边长为：$B/2$，直角边长为：$B/2\sqrt{2}$ ；

平行四边形的两边长分别为 $B/2$、$B/2\sqrt{2}$ ；

正方形的边长为：$B/2\sqrt{2}$ 。

3. 基本图块过程的编写

我们先来编写三角形、平性四边形和正方形三个基本图形的绘制过程。

(1) 画三角形的过程为“A”

```
TO A X
  MAKE "Z :X/SQRT 2
  FD :Z RT 135 FD :X RT 135
  FD :Z RT 90
END
```

(2)画平行四边形的过程为“C”

```
TO C X
  MAKE "Z :X/SQRT 2
  REPEAT 2 [FD :Z RT 45 FD :X RT 135]
END
```

(3)画正方形的过程为“D”

```
TO D X
  REPEAT 4 [FD :X RT 90]
END
```

4. 主过程的编写

“QB”为主过程，通过调用前三个过程画出如图9—6所示的七巧板结构图。过程清单如下：

```
TO QB B
  MAKE "B1 :B/SQRT 2
  MAKE "B2 :B/2
  HT LT 90 A :B1 FD :B RT 135
  C :B/2
  FD :B1 RT 90 A :B/2 LT 90 D :B2
  FD :B2 A :B/2 BK :B2 LT 90 A :B
  LT 90 A :B
END
```

5. 白鹅

现在我们利用七巧板来拼凑一只白鹅，过程清单如下：

```
TO E B
  MAKE "B1 :B/SQRT 2
  MAKE "B2 :B1/2
  HT PU BK 30 PD RT 135 A :B/2 RT 135
  A :B RT 90 FD :B1 RT 90 A :B
  FD :B/7 LT 90 A :B1 FD :B/2 LT 45
  BK :B2 C :B/2 RT 45 FD :B/2 LT 45
  D :B2 RT 90 FD :B2 LT 90 A :B/2
END
```

执行命令"E 60"将画出如图9－7所示的白鹅。

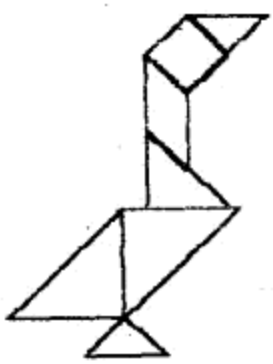


图9－7　白鹅

6. 毛驴

下面再来拼凑一头小毛驴，过程清单如下：

```
TO LU B
  MAKE "B1 :B/SQRT 2
  MAKE "B2 :B1/2
```

```
  HT LT 90 C :B/2 LT 180 A :B FD :B1
  LT 45 FD :B1 D :B2 LT 90 A :B/2
  RT 90 FD :B2 RT 180 A :B/2 FD :B2
  LT 45 FD :B1 A :B1 RT 90 A :B
END
```

执行命令“LU 60”将画出如图9－8所示的毛驴。

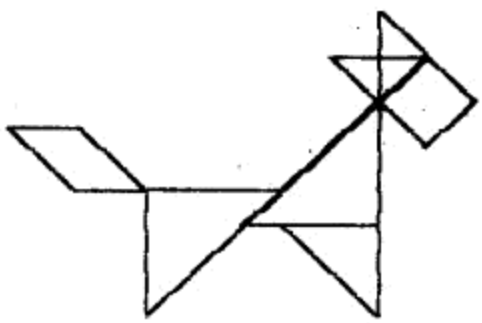


图9－8

7. 猫咪

再来拼凑一只小猫咪，过程为“MAO”。

```
TO MAO B
  MAKE "B1 :B/SQRT 2
  MAKE "B2 :B1/2
  HT PU FD :B/2 PD RT 45 A :B/2 RT 180
  A :B/2 FD :B2 LT 90 FD :B2 LT 90
  BK :B2 A :B FD :B1 RT 90 FD :B1
  RT 90 A :B LT 45 FD :B/2 RT 90
  A :B1 RT 120 C :B/2
END
```

执行命令“ MAO 60 ”，将画出如图9－9所示的小猫咪。相

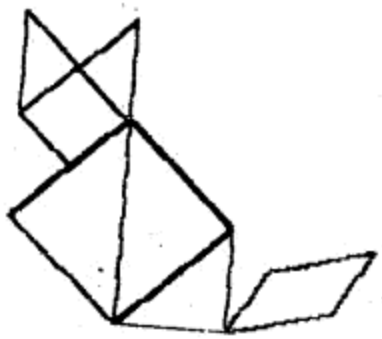


图9—9

信，由三个拼图的示范，读者定能拼出更精彩的图形。

第十章　字表处理

现在，我们已经学会了怎样控制海龟在屏幕上画出各种各样的图形。其实，LOGO 语言的作用并不是单纯的画图，它还能写诗、作文及解数学题，还能创作出有趣的游戏，真是你的好伙伴呢！。

LOGO 写诗、解题、做游戏的过程，就是利用屏幕和键盘进行人、机对话的过程。为了实现人与电脑之间的理解和交流，首先必须学会使用 LOGO 的字和句，电脑也同样以字和句的方式回答你的问题。这样，我们不仅可以用图形、还可以用句子与电脑交流，并从中了解电脑的工作过程。

电脑能够处理的各种信息（包括数、字、句子、图形等）称为数据。在 LOGO 语言中，除图形外，经常处理的数据还有：数、字、表。

本章主要介绍 LOGO 的字、表处理，同时讨论下列指令：

前取字指令	FIRST
前舍字指令	BF
后取字指令	LAST
后舍字指令	BL
连字指令	WORD
表前取元素指令	FIRST
表后取元素指令	LAST
表前舍元素指令	BF
表后舍元素指令	BL

表前插入元素指令	FPUT
表后插入元素指令	LPUT
表合并指令	LIST
表连接指令	SE
从键盘读入一个字符指令	RC
从键盘读入多个字符指令	RQ
条件测试指令	TEST
条件成立执行指令	IFT
条件不成立执行指令	IFF
字检测指令	WORD?
数检测指令	NUMBER?
表检测指令	LIST?
海龟状态检测指令	TS
开跟踪指令	TRACE
关跟踪指令	NOTRACE

第一节　字与字运算

一、LOGO 中的字

在 LOGO 语言中，“字”可以是字母、数字，也可以是由字母、数字、符号(不包括运算符号和括号)组成的不带空格的字符串。例如：

"AA

"QUIET!

"WHO?

"2PROM

"2,A(—

"HOUR:MINU:SECO

等都是字。

为了显示在人机对话时，LOGO 输出的字、句，就需要使用打印显示命令“PR”和“PRINT1”。上一章已经对这两条命令的使用进行了简单的介绍，本章将继续介绍这两条命令打印显示字、句等内容的使用方法。

在 PR 语句中，为了区别命令名和字，规定字的前面必须加上双撇号“ " ”以作为字的标记符，因此也可以称它为“带撇号的字”或“LOGO 字”。例如：键入

PR FD 50

命令时，因“FD”前面没有双撇号，LOGO 把它作为命令处理，执行的结果是使海龟向前移动 50 步。如果在“FD”的前面加上双撇号，海龟就不会移动了。键入

PR "FD 50

将显示

FD

RESULT 50

输出结果之所以产生差别，是由于系统把“ "FD ”当做 LOGO 字进行处理，原样输出“FD”，“50”则作为一个结果输出。

双撇号后面没有任何字符时也是一个字，这样的字称为“空字”，它可产生一种特殊的打印效果。例如：键入

PR "

将产生一个空行。

字的后面不能加双撇号。如果字的后面加上双撇号时，LOGO 将把双撇号也作为字进行处理。例如：键入

PR "GOOD"

将显示

GOOD"

在打印显示字的时候，最容易疏忽的地方就是在字的前面漏掉双撇号。例如：键入

PR GOOD

电脑并没有显示 GOOD 字样，反而显示错误信息，它告诉你，在 LOGO 语言中没有叫 GOOD 的过程名。显然，因 GOOD 前面没有加双撇号，电脑把它当做命令或过程名处理，由于没有定义 GOOD 过程，所以显示出错信息。

二、字的运算

LOGO 语言提供了五条关于字运算的命令，四条用于字的分解，一条用于字的连接。字的运算结果仍然是字。

1. **前取字指令** FIRST

格式：FIRST 参数

类型：语句、命令

功用：取出字的第一个字符形成一个单字符的字。

注释：FIRST 在英文中是“第一”的意思。

说明：参数可以是“字”，也可以是赋过值的变量。

示例：键入

PR FIRST "ABCD　　（回车后，将显示

A

输出结果表明，仅取“ABCD”中的第一个字符。

2. **前舍字指令** BF

格式：BF 参数

类型：语句、命令

功用:舍去字的第一个字符,由余下的字符组成新的字。

注释:BF 来源于"But First",是"不是最前"的意思,在此表示"由除第一个之外的有字符组成新字"。

说明:参数可以是"字",也可以是赋过值的变量。BF 函数的作用就是输出这个"参数"中除第一个字符以外的"字"。

示例:键入

PR BF "ABCD　　(回车后,将显示

BCD

输出结果舍去了"ABCD"中的第一个字符"A"。

3. **后取字指令** LAST

格式:LAST 参数

类型:语句、命令

功用:取出字的最后一个字符组成一个单字符的新字。

注释:LAST 在英文中为"最后"的意思。

说明:参数可以是字,也可以是赋过值的变量。LAST 仅输出这个字的最后一个字符。

示例:键入

? PR LAST "ABCD　　(回车后,将显示

D

表示 LAST 取出了字"ABCD"的最后一个字符"D"。

4. **后舍字指令** BL

格式:BL 参数

类型:语句、命令

功用:舍去字的最后一个字符,由余下字符组成新字。

注释:BL 来源于"But Last",是"不是最后"的意思,在此表示"由除最后一个字符以外的所有字符组成新字"。

说明:参数可以是字,也可以是赋过值的变量。输出这个

字中除最后一个字符以外的所余字符。

示例：

？PR BL "ABCD　　（回车后，将显示

ABC

执行结果中舍去了“ABCD”中的最后字符“D”。

5. **连字指令** WORD

格式：WORD 参数 1 参数 2

类型：语句、命令

功用：把参数 1 和参数 2 连接起来组成一个新字。

注释：WORD 在英文中是“字”的意思。在此表示把后面的两个字连接起来组成一个新字。

说明：参数 1 和参数 2 可以是具体的字，也可以是赋过值的变量。

示例：

？PR WORD "FOOD "BALL　　（回车后，将显示

FOODBALL

结果表明，WORD 函数把“FOOD、BALL”两个连接起来，组成了新字“FOODBALL”。

三、“数”等于“字”

LOGO 中的“数”就是“字”。也就是说：数据不必冠以双撇号，就可以作为 LOGO 字处理，LOGO 的所有字运算指令对数同样有效。例如：键入

？PR FIRST 4321　　（回车后，将显示

4

再键入

？PR BF 4321　　（回车后，将显示

321

算术运算和字运算可以同时用在一个表达式中，例如：键入

? PR (WORD 12 34)+(WORD 56 78)　　(回车后，将显示

6912

它表示，首先把“12、34”两个字连接起来组成新字“1234”、把“56、78”两字连接起来组成“5678”；然后，对两个新字求和“1234+5678=6912”。

若键入

? PR (BL 5678)+(BF 8765)　　(回车后，将显示

1332

执行结果表示：首先舍去“5678”中的最后一个字符“8”得到新字“567”；然后舍去“8765”中的第一个字符“8”得到新字“765”；最后进行“567+765”的求和运算。

四、“正”字计数

我国古代，经常使用“正”字的笔划进行计数。现在用电脑模拟该种计数方法。要求输入一个不超过 40 的自然数，在屏幕上用画“正”字的形式表示这个数。过程清单如下。

```
TO A1
  HT RT 90 FD 36 BK 18 LT 90
END

TO A2
  A1 BK 50 FD 25
END
```

```
TO A3
  A2 RT 90 FD 16 BK 16 LT 90 BK 25
END

TO A4
  A3 LT 90 PU FD 12 RT 90 PD FD 25
  BK 25 RT 90 PD
END

TO A
  A4 BK 8 FD 40 BK 20 LT 90 FD 50
END

TO Z L
  REPEAT :L [A RT 90 PU FD 40 PD LT 90]
END

TO XY X Y
  PU SETXY :X :Y PD
END

TO XZ S
  IF ANYOF :S<1 :S>40 STOP
  MAKE "M QUOTIENT :S 5
  MAKE "N REMAINDER :S 5
  TEST :S>20
```

```
    XY -120 80
    IFT Z 4 XY -120 0 IF :N=0 Z :M-4 ELSE Z :M
-4 P
    IFF IF :N=0 Z :M ELSE Z :M P
  END

  TO P
    IF WORD "A :N="A1 A1
    IF WORD "A :N="A2 A2
    IF WORD "A :N="A3 A3
    IF WORD "A :N="A4 A4
  END
```

程序中：

A1、A2、A3、A4、A 分别是画一笔、两笔、三笔、四笔、五笔的过程(见图 10—1)。

图10--1 “正”的各笔划

“Z”过程控制画 L 个“正”字，“XY”过程控制海龟的坐标。

“XZ”是主过程。S 被 5 除所得的商 M 是全“正”字的个数；余数 N 是零笔道数。S>20 时分两排画“正”字，先在上排画四个，然后海龟移到下一排的起始处。若余数 N=0，则只画

剩下的 M－4 个“正”字；若余数不为 0，先画出 M－4 个“正”字；再将 A 与 N 连接起来组成一个字，这个字恰好就是画 N 笔的过程名；然后再调用“P”过程进行判断。如：当余数是 3 时则调用过程“A3”，通过“P”过程调用“A3”过程，画出余下的笔划。S≤20 的画法类似(在一行内画)。

如执行“XZ 9 ”，将画出图 10－2 所示的图形。

图10－2　计数为9时画的“正”字

注——过程“XZ”中使用了条件测试指令“IFT、IFF”命令，具体功能将在第四节讨论。

第二节　表与表运算

一、LOGO 中的表

1. 表的概念

我们知道，若干个汉字可以组成句子，若干个句子可以组成文章。LOGO 语言既然有“字”，它是否也能组成“句子”呢？它的“句子”取什么形式呢？

LOGO 中的“句子”叫做“表”。若干个“数”或不带引号的“字”连在一起，再用方括号将它们括起来，就组成了 LOGO 中的句子——“表”。

表中的字和数叫做表的“元素”，各元素之间用空格隔开。另外，表还可以套表，也就是说“表”也可以作“元素”。

2. 表的形式

LOGO 中的表有多种形式，下列的五行字串都是表。

[1 2 3 4 5]

[3A 0.6 D+E CC;]

[HOW ARE YOU?]

[= 1.2N4 3A./[A BB C;]]

[[HAO HAO XUE XI][TIAN TIAN XIANG SHANG]]

“表”所表示的内容很丰富。因 LOGO 语言中没有数组，必要时可以用表来代替数组。

我们可以用 PR 语句来输出一个表。电脑在输出表时将自动去掉最外层的括号，这样就更像日常生活中的句子了。例如：

键入

? PR [HOW DO YOU DO]（回车后，将显示

HOW DO YOU DO

键入

? PR [THE PEN IS [RED]]（回车后，将显示

THE PEN IS [RED]

3. 注意事项

使用表时，要注意左、右括号的配对。

只含一个字的表如“[A]”和字“"A”是两种量（尽管它们用 PR 命令时，显示相同的结果）

如果方括号内没有任何东西，或者说只有空格，那么这是一个空表，可以记作“[]”。用 PR 命令显示空表，如同空字一样，会产生一个空行。

空表常常是表运算的结果，编程中常用检查结果是否为

空表的手段来控制程序的执行。下述两个运算的结果都是空表：

BF [A]　　(回车后，将显示

RESULT：[ ]

BL [A]　　(回车后，将显示

RESULT：[ ]

空表和空字是两回事，如果键入：

[]=″

将显示

RESULT：FALSE

表明空字等于空表的判断是错误的。

二、表的有关运算

LOGO 语言提供了八条关于表运算的命令，上节用于分解字的四个运算同样也可以用于分解表，除了 FIRST、LAST 作用于表的结果可能是字或表之外，其它运算的结果仍然是表。下面分别介绍：

1. **表前取元素指令** FIRST

格式：FIRST 参数

类型：语句、命令

功用：取出表的第一个元素。

说明：参数可以是字或表，也可以是赋过值的变量。参数是表时将输出表的第一个元素。

示例：键入

PR FIRST [THIS IS A LIST]　　(回车后，将显示

THIS

2. **表后取元素指令** LAST

格式:LAST 参数

类型:语句、命令

功用:取出表的最后一个元素。

说明:参数可以是字或表,也可以是赋过值的变量。参数是表时将输出这个表的最后一个元素。

示例:键入

PR LAST [THIS PEN IS [RED]] (回车后,将显示

RED

键入

PR LAST [THIS PEN IS RED] (回车后,将显示

RED

上述两条命令虽然显示结果的形式相同(都显示 RED),但它们有质的区别。如键入下述的等式:

LAST [THIS PEN IS [RED]]=LAST [THIS PENIS RED]

将显示

RESULT: FALSE

这表明上面的等式是不成立的,等式左边输出的结果是表 [RED],等式右边输出的结果是字"RED。

3. **表前舍元素指令** BF

格式:BF 参数

类型:语句、命令

功用:舍去表的第一个元素,由余下元素组成新表。

说明:参数可以是字或表,参数是表时输出一个新表,这个表是参数删除第一个元素后得到的。

示例:键入

PR BF [THIS IS A LIST] (回车后,将显示

IS A LIST

4. 表后舍元素指令 BL

格式:BL 参数

类型:语句、命令

功用:舍去表的最后一个元素,由余下元素组成新表。

说明:参数可以是字或表,参数是表时输出的仍然是表,这个表是参数删除最后一个元素后得到的。

示例:键入

PR BL [THIS IS A LIST] (回车后,将显示

THIS IS A

5. 表前插入元素指令 FPUT

格式:FPUT 参数 1 参数 2

类型:语句、命令

功用:将参数 1 插入参数 2 的最前面,组成新表,两参数之间自动插入一个空格。

注释:FPUT 来源于"First PUT",是"前面放置"的意思,在此表示"从表的前面插入元素"。

说明:参数 2 必须是一个表,参数 1 必须是能够充当表的元素,输出仍然是一个表。新表所含元素的个数,为参数 2 元素个数加一。

示例:键入

PR FPUT [AB] [C D]　　(回车后,将显示

[AB] C D

6. 表后插入元素指令 LPUT

格式:LPUT 参数 1 参数 2

类型:语句、命令

功用:将参数 1 插入参数 2 的最后面,组成新表,两参数

之间自动插入一个空格。

注释:LPUT 来源于"Last PUT",是"后面放置"的意思。

说明:参数 2 必须是一个表,参数 1 必须是能够充当表的元素。输出仍然是一个表,新表所含的元素个数,为参数 2 所含元素的个数加一。

示例:键入

PR LPUT [AB] [C D]　　(回车后,将显示

C D [AB]

键入

PR LPUT "Z [X Y]　　(回车后,将显示

X Y Z

7. **表合并指令** LIST

格式:LIST 参数 1 参数 2

类型:语句、类型

功用:将两个元素合并,组成一个新表。

注释:LIST 在英文中为"表"的意思。在此表示"两表合并组成新表"。

说明:两个参数都可以是字或表,输出是一个新表。新表的元素就是这两个参数。

示例:键入

PR LIST "A "B　　(回车后,将显示

A B

键入

PR LIST [YUAN] [SU]　(回车后,将显示

[YUAN] [SU]

8. **表连接指令** SE

格式:SE 参数 1 参数 2

类型:语句、指令

功用:把参数 1、参数 2 连接起来,组成一个新表。

说明:参数 1、参数 2 可以是一个具体的字或表,也可以是赋过值的变量。把字或表连接后中间自动加一空格,元素如是表则把中括号去掉。

示例:键入

```
PR SE "WO [AI LOGO]    (回车后,将显示
WO AI LOGO
```

键入

```
PR SE "WO "AI          (回车后,将显示
WO AI
```

键入

```
PR SE [WO AI],[LOGO!]  (回车后,将显示
WO AI LOGO!
```

第三节　键盘扫描

为了方便进行人机对话,LOGO 语言中设置了两个键盘扫描指令"RC、RQ"。

一、从键盘读入一个字符指令 RC

格式:RC

类型:语句、命令

功用:等待按键,从键盘读取一个字符传给使用它的操作命令,操作命令一般为 MAKE 或 PR。读入字符不须按回车键,输入的字符不在屏幕上显示。

注释:RC 来源于"Read Character",是"读入字符"的意

思。它的功能相当于 F BASIC 的“INKEY $ (0)”指令。

RC 命令从键盘读入字符时不须按回车键。

说明：如果按数字键，则输入一个数值，它可以直接参加数字运算。如果按字符键，则表示输入的是一个字，它可以参加字的运算。

示例：

```
TO A
  MAKE "X RC
  PR 5+:X
END
```

执行过程“A”，当运行到 RC 时暂停，等待用户使用键盘输入，如果按数字键“8”，则把“8”赋值给变量 X，继续执行后，就显示“5+X”的值 13。

再如：

```
TO INST
  COMM
  INST
END

TO COMM
  MAKE "X RC
  IF :X="F FD 10
  IF :X="R RT 30
  IF :X="L LT 30
  IF :X="C CS
  IF :X="Q TOPLEVEL
END
```

执行“INST”过程，可以使用规定的按键控制海龟绘图。按“F”键，海龟前进 10 步；按“R”键，海龟右转 30 度；按“L ”键，海龟左转 30 度；按“C”键，清屏幕；按“Q”键，中止过程运行。

对刚开始学习 LOGO 语言绘图的小朋友，过程“INST”将成为他(她)最好的伙伴。

二、从键盘读入多个字符指令 RQ

格式：RQ

类型：语句、命令

功用：等待用户输入一行字符，传送给使用它的那个操作命令。这个命令一般是 MAKE 或 PR。所键入的内容将显示在屏幕上，并以回车作为字符的结束。

注释：RQ 来源于“ReQuest”，是“要求、请求”的意思，在此表示“要求从键盘输入字符”。

RQ 命令相当于 F BAISC 中的“INPUT”指令。

说明：RQ 所转送的是“表”形式数据，但键入的数据两边不需要方括号。

示例：

```
? MAKE "X RQ
YX XING                    (键入数据
? PR :X                    (回车后，将显示
YU XING
```

再如：

```
TO B
  MAKE "X RQ
  PR :X
```

END

执行过程“B”。当运行到 RQ 时暂停，等待用户从键盘输入一行字符。如果键入：

A＊B＝12

则显示：

A＊B＝12

第四节　测试与检测

一、测试

LOGO 的条件测试指令有三个，它们实际上是条件命令的另一种形式。

1. **条件测试指令** TEST

格式：TEST 条件

类型：语句、命令

功用：对条件进行测试，条件成立为真(TRUE)，条件不成立为假(FALSE)。

注释：TEST 在英文中为“测试”的意思。

说明：TEST 命令常与 IFT、IFF 两个语句配套使用，测试结果由它们取出。

2. **条件成立执行指令** IFT

格式：IFT 执行语句 1

类型：语句、命令

功用：根据最近一次 TEST 命令的测试结果，若结果为真(TRUE)则执行语句 1。

注释：IFT 来源于“IF True”，是“如果为真”的意思。

说明：如果最近一次测试结果为假(FALSE)时，则跳过语句1。

3. **条件不成立执行指令** IFF

格式：IFF 执行语句 2

类型：语句、命令

功用：根据最近一次 TEST 命令的测试结果，若结果为假(FALSE)，则执行语句 2。

注释：IFF 来源于"IF False"，是"如果为假"的意思。

说明：如果最近一次测试结果为真(TRUE)时，则跳过语句 2。

上述三个语句联合使用很像条件语句"IF THEN ELSE"，但与条件语句有差别。条件语句是一个语句整体，条件成立与否的结果要立即使用。而 TEST 的测试结果可以不马上用，直到下一次测试之前，测试结果一直有效，可以多次使用。另外，条件的测试、判断分开，可避免条件语句过长的情况。

下面是一个加减法练习过程。

```
TO T N X S
IF :X=:N PRINT1 "S= PR :S/:N * 100 STOP
MAKE "A 1+RANDOM 2
MAKE "B 10+RANDOM 90
MAKE "C 10+RANDOM 90
TEST :A=1
IFT PRINT1 :B PRINT1 "+ PRINT1 :C PRINT1 "=
IFF PRINT1 :B PRINT1 "- PRINT1 :C PRINT1 "=
MAKE "D FIRST RQ
IFT PRINT1 :B + :C = :D GO "E
```

```
IFF PRINT1 :B - :C = :D E:
IFT MAKE "S :S+1 PR [DUI LE]
IFF PR [CUO LE]
T :N :X+1 :S
END
```

这是一个递归过程，参数 N 的初始值是练习题数；X 累计题数，初值为 0，X＝N 时结束练习；S 是累计正确的题数，初值为 0。A 的取值随机产生(1、2)，A＝1 出加法题，A＝2 出减法题。如果你想做 5 道题，则执行“T 5 0 0”。做题结束由电脑自动评分。

二、检测

1. **字检测指令** WORD?

格式：WORD? 参数

类型：语句、命令

功用：如果 WORD? 后面参数是字，则输出为真(TRUE)，否则为假(FALSE)。

注释：WORD? 的英文含义是“是字吗?”。

说明：WORD? 后面的参数可以是变量、字符串、数字等。

示例：

```
TO C A
  PR WORD? :A
END
```

执行“C R”，将显示“TRUE”，表示当前变量 A 中的内容是字。

执行“C 5＋4”，将显示“TRUE”，表示变量 A 中的内容是字。

执行“C [1 2 3]”，将显示“FALSE”，表示变量 A 中的内容不是字(是表)。

2. 数检测指令 NUMBER?

格式：NUMBER? 参数

类型：语句、命令

功用：如果 NUMBER? 后面的内容是数字，则输出为真(TRUE)，否则为假(FALSE)。

注释：NUMBER? 的英文含义是“是数吗”。

说明：NUMBER? 后面的参数可以是变量、字符串、数字等。

示例：

```
TO C A
  PR NUMBER? :A
END
```

执行“C 1234”，将显示“TRUE”，表示变量 A 中的内容是数字。

执行“C "R”，将显示“FALSE”，表示变量 A 中的内容不是数字。

3. 表检测指令 LIST?

格式：LIST? 参数

类型：语句、命令

功用：如果 LIST? 后面的内容是表，则输出为真(TRUE)，否则为假(FALSE)。

示例：

```
TO C A
  PR LIST? :A
END
```

执行“C "R”后，将显示“FALSE”，表示变量 A 中的内容不是表。

执行“C [1 2 3]”后，将显示“TRUE”，表示变量 A 中的内容是表。

4. 海龟状态检测指令 TS

格式：TS

类型：语句、命令

功能：输出海龟当前的状态。

注释：TS 来源于“Turtle Status”，表示“测试海龟状态”。

说明：执行 TS 命令将输出两项状态。第一项为笔状态，落笔为真(TRUE)，提笔为假(FALSE)；第二项为海龟显示状态，显龟为真(TRUE)，藏龟为假(FALSE)。

示例：

? HT TS　　(回车后，显示

RESULT：[TRUE FALSE]

测试结果表明，当前处于落笔、藏龟状态。

三、跟踪命令

1. 开跟踪指令 TRACE

格式：TRACE

类型：命令

功用：打开跟踪开关，逐行跟踪过程的执行。

注释：TRACE 在英文中为“跟踪、追踪”的意思。

打开跟踪开关后，每读到一条命令时，将显示该命令的内容，等待用户按任意键后才执行该命令。

说明：跟踪命令在程序运行时，能显示、打印它的运行线路和变量变化情况。通过运行跟踪，监视程序的动态趋向，很

容易确定程序中存在的问题和位置。是调试程序的得力工具。

示例:跟踪一个平方表过程的执行。

```
TO PFB A B
  IF :A=:B STOP
  PRINT1 :A PRINT1 "^  :A * :A
  A :A+1 :B
END
```

执行过程“PFB 1 10”,屏幕将显示:

```
1^ 1
PROCEDURE NO FOUND IN LINE A :A+1 "B
? ■
```

这个过程的错误是非常明显的,在这里,为了演示TRACE 命令的使用,我们故意将递归过程 END 前一行的“PFB”写成“A”。这就必然显示错误信息。现在察看怎样用TRACE 命令追踪程序运行。

```
键入
? TRACE                    (回车后,显示
TRACE ON                   (表示跟踪开关打开
键入
? PFB 1 10                 (回车后,显示
EXECUTING PFB              (正在执行
IF :A=:B STOP■             (显示将执行的命令
■为闪烁的光标,当代用户按任意键。(按任意键后
PRINT1 :A PRINT1 "^  PR :A * :A■
1 PRINT1 "^  PR :A * :A■
^  PR :A * :A■
A :A+1 :B■
```

PROCEDURE ON FOUND IN LINE: A :A+1 :B

? ■

提示信息说明:没有发现过程名是 A 的过程。

2. **关跟踪指令** NOTRACE

格式:NOTRACE

类型:命令

功用:撤消跟踪功能。

注释:NOTRACE 在英文中为"不跟踪"的意思。

说明:TRACE、NOTRACE 命令一般成对使用,使用 TRACE 命令打开跟踪后,若不执行 NOTRACE 命令撤消,跟踪一直有效。

示例:上例中键入

? NOTRACE　　(回车后,将显示

TRACING OFF　　(关闭跟踪开关

以上,我们讨论的是在命令状态下使用跟踪命令,它也可以像其它命令一样使用在程序中。当在过程中执行开跟踪命令时,并不暂停等待击键逐行执行命令,而是每运行一行程序,便显示一行信息。根据提示信息,用户可以详细了解程序的运行线路和变量的变化情况。

第五节　变量的深入讨论

一、变量的赋值

我们已经有了"参数"的概念,而且也知道参数也叫做"变量",它们是同一概念的两个名称。在使用带参数的过程控制海龟作图时,赋给过程的实际参数通常是一个确定的数值常数。实际上,参数赋值也可以是字或表。下面我们先写一个过

程，然后给变量分别赋予常数、字、表，看一看它的输出结果。

```
TO SU X
  PR SE [WO SHUO] :X
END
```

分别给“SU”过程的变量 X 赋予数、字、表：

键入“SU 1234”，将显示“WO SHUO 1234”；

键入“SU "NIHAO”，将显示“WO SHUO NIHAO”；

键入“SU [WO AI LOGO]”，将显示“WO SHUO WO AI LOGO”。

在“SU”过程里，你还可以输入其它的数、字、表，看看会显示出什么结果。

二、变量的赋值语句

过程内对变量赋值的语句“MAKE "X Y”，也可以用来对变量赋字或表。它有两个输入项，第一项是"X，用来作为赋值语句的变量名("X 是一个字，LOGO 语言规定：赋值语句的第一项，即变量，必须使用 LOGO 字)；第二项 Y，它是将要赋给变量"X 的“值”，这个值可以是数、字、或者表。

例如：键入

```
? MAKE "X (-1234)
? PR :X
```

将显示

```
-1234
```

键入

```
? MAKE "X "NIHAO
? PR :X
```

将显示

NIHAO

键入

? MAKE "X [ZHE SHI BIAO]

? PR :X

将显示

ZHE SHI BIAO

注意：如果对一个变量名使用两次 MAKE 命令，那么前面的一次赋值将被后面的赋值所代替。例如：键入

? MAKE "X [ZHE SHI BIAO]

? MAKE "X "NIHAO

? PR :X

将显示

NIHAO

三、变量的形式

LOGO 的变量有双撇号变量、冒号变量两种形式，它们是为适用不同场合而设置的。当变量被赋值时，使用双撇号变量名；在其它状态下一般使用冒号变量名。LOGO 中的"字"只有在 MAKE 命令之后作为第一个输入参数时，才代表一个变量；在其它情况下的字均不代表变量。

四、赋值语句 MAKE 的使用

MAKE 命令与其它命令一起使用，可以构成各种复杂的句子。例如：

MAKE "Y [ZHE SHI BIAO]

PR :Y

MAKE "Y SE "NIHAO :Y

PR :Y

MAKE "Y SE BUTFIRST :Y [WO AI TA]

PR :Y

现在分析上述 MAKE 命令的工作情况。首先键入：

MAKE "Y [ZHE SHI BIAO]

PR :Y

这时，MAKE 建立了名字叫 Y 的变量，并对变量 Y 赋值，这个值就是“ZHE SHI BIAO”。“PR :Y”进行取值显示，此时的结果是：

ZHE SHI BIAO

再继续键入

MAKE "Y SE "NIHAO :Y

PR :Y

在这一条赋值语句中，执行的顺序是先执行“SE "NIHAO :Y ”，就是说，先将“NIHAO”和“Y”合成为一个新字或者表，而这时 Y 的内容是“[ZHE SHI BIAO]”，所以合成的新表就是“[NIHAO ZHE SHI BIAO]”。新表合成后，MAKE 又把它赋给变量 Y，由于已经有了一个变量 Y，所以，此时 MAKE 不是重新建立 Y 变量，而是给变量 Y 重新赋值，这个新值就是刚合成的新表。随后，“PR :Y”又显示变量中的新内容。

NIHAO ZHE SHI BIAO

如果再继续键入：

MAKE "Y SE BUTFIRST :Y [WO AI TA]

PR :Y

上面这条赋值语句，首先执行的是“BUTFIRST :Y”。而这时的 Y 已经取了新值[NIHAO ZHESHI BIAO]，由于

BUTFIRST 是选取除去表中第一个元素以外的元素组成新表，所以这个合成命令实际上是将[ZHE SHI BIAO]和[WO AI TA] 合成一个新表 [ZHE SHI BIAO WO AI TA]。紧接着 MAKE 又一次给 Y 变量赋新值。“PR :Y”显示 Y 变量的新值：

ZHE SHI BIAO WO AI TA

五、使用 MAKE 指令的注意事项

以上，我们较详细地介绍了赋值语句与其它命令连用的例子，对于这一类连续使用赋值语句的情况，要特别注意变量名的形式不能搞错，否则将出现一些意想不到的结果。例如：

键入

```
MAKE "X "WANG
PR :X
```

显示：

WANG

再键入：

```
MAKE :X [ZHE SHI TA!]
PR :X
```

显示：

WANG

再键入：

```
PR :WANG
```

显示：

ZHE SHI TA!

上例中，最后两次显示的结果可能出乎你的预料。

下面分析产生这个结果的原因。

1. 执行第一个 MAKE 语句，"WANG 赋给了变量"X，"PR :X"就变成了：

PR "WANG

因此显示"WANG"。

2. 执行第二个 MAKE 语句，

MAKE :X [ZHE SHI TA!]

由于在第一个 MAKE 语句中，已经把"WANG"赋值给变量 X，现在对冒号变量形式的 X 赋值，就相当于对变量 WANG 赋值，这样一来，第二条赋值语句实际上就变成了以"WANG"为第一个输入变量的赋值语句，即

MAKE "WANG [ZHE SHI TA!]

因此，当执行

PR :X

命令时，将显示变量 X 的内容，当执行

PR :WANG

命令时，将显示变量 WANG 中的内容。即

ZHE SHI TA!

从上面这个例子可以看出，使用变量名时，必须特别注意两种变量名形式的使用。否则，将会产生意想不到的后果。

第六节　字和表的应用

一、字的应用——颠倒字的排列顺序

1. 编程思路

对于字"ABCD"，假如要把它变成"DCBA"，可以这样处理：把除最后一个字符"D"以外的其他字符顺序颠倒（即把 ABC 变成 CBA），再取出最后一个字符 D，放在已颠倒顺序字

符 CBA 的前面，就实现了字 ABCD 排列顺序的颠倒。

为什么要进行这样的处理呢？

2. 编程原理

这样做可以把颠倒一个较长字的处理，简化为一个较短字的处理。为此，用 BL 命令去掉 ABCD 的最后一个字符 D，再对余下的短字 ABC 继续作上面的处理，即 BL 的 BL 操作。这样，字就越来越短。整个变化过程如下：

REVE "ABCD=D—(REVE "ABC)

→ D—C—(REVE "AB)

→ D—C—B—(REVE "A)

→ D—C—B—A

颠倒字变成颠倒该字的 BL 部分，反复进行直至该字为空停止。

3. 最初过程

根据以上讨论，导出下列递归过程：

```
TO REVE X
  OP WORD LAST :X REVE BL :X
END
```

当执行该过程时，电脑并不能正常工作，显示出错信息。出错的原因并不难找，因为当字减为空字时，我们并没有让过程停下来。过程试图对一个空字作 LAST 操作，当然要出错误。

4. 过程修正

为了解决这个错误，应该在过程中加一判断语句，当执行 REVE 操作到空字时，直接输出最后字符。修改后的过程如下：

```
TO REVE X
```

```
    IF :X=" OP :X
    OP WORD LAST :X REVE BL :X
  END
```

如果键入：

```
  ? REVE "ABCDEF
```

将显示：

```
  RESULT: FEDCBA
```

二、表的应用

表的应用非常广泛，如用表描述电话号码本；一个学校各班级的编号和学生人数；一个单位的职工名单和他们的工资；学生各科成绩的总平均成绩单；银行存款单号码和储户的名单；学习写作、造句等等。由于篇幅的限制就不一一举例，在这里只介绍一个数列的例题，希望能给读者带来一定的帮助。

1. 费氏数列的概念

学习数学时，常见到一些数列，如：等差数列，等比数列等等。在这里我们要介绍另一种形式的数列。请看下面一组数：

1 1 2 3 5 8 13 21 34 55……

这组数的规律是，每个数都是其前面两个数的和。例如：

8＝3＋5

5＝2＋3

等。具有这种性质的数列就是“费氏数列”。

2. 编程思路

假如把数列看成是一个表，且表中的元素对应于数列中的每一项，实际上就是把处理数列问题转换成 LOGO 语言中的字表处理问题。

根据“费氏数列”每个数都是其前面两个数之和的定义，

我们只要截取表中最后一个元素(可以通过 LAST 操作获得)、最后第二个元素(可以对经过 BL 操作后留下的表,再进行 LAST 操作获得),然后把截取的这两个数相加,即可得到一个新数。如果把新得到的数加到数列的末尾,或者说把该数作为一个元素,添加在最后,费氏数列就得到了扩展。

3. 编写基本过程

下面是一个产生"费氏数列"新数的过程:

```
TO XIN S
  MAKE "X (LAST :S)+(LAST BL :S)
  MAKE "S SE :S :X
  OP :S
END
```

这个过程的输入值是一个表,通过截取表中最后两个元素并相加获得一个新的数值。再通过 SE 命令把新的数值添加到表的末尾,即组成一个新的数列。例如:

输入:

```
PR XIN [2 3]
```

将显示:

```
2 3 5
```

运行结果说明,原数列增加了一项,产生了一个新的数列。

5. 增加 N 项费氏序列数的过程

如果要增加 N 项费氏序列数,仅靠上面的过程一遍遍地去做,那就太笨拙了。使用重复命令 REPEAT 可解决这一问题。编写过程如下:

```
TO SHU N
  MAKE "S [1 2]
  REPEAT :N [MAKE "S XIN :S]
```

```
    PR :S
  END
```

其中 N 为需要增加的项数，例如：键入

```
  SHU 8
```

将显示：

```
  1 2 3 5 8 13 21 34 55
```

键入：

```
  SHU 13
```

显示：

```
  1 2 3 5 8 13 21 34 55 89 144 233
  377 610
```

5. 费氏序列与黄金分割法

运用字表处理原理产生费氏数列并不是我们的唯一目的。费氏数列除了定义中的规律外，还有其它的特点和性质。

让我们先来做个试验，拿上面给出的费氏数列的最后一项去除倒数第二项，即 377/610，看看会出现什么结果？

键入：

```
  PR 377/610
```

将显示：

```
  .61803
```

正好近似于黄金分割比值 0.618。再试一次其它的情况，看看是否也有这个特点。假如现在的数列只有 10 项，那么最后一项除倒数第二项应为 34/55。

键入：

```
  PR 34/55
```

将显示：

```
  .61818
```

结果与黄金分割值仍然十分近似。

当然，并不是所有的相邻两项都满足这一特性，比如最前面几项，相邻两项的比值就不是黄金分割比例值了。那么，费氏数列项数的多少与黄金分割比例又有什么关系呢？实践证明：费氏数列的项数越多，其最后两项之比就越接近于黄金分割比例值。

刚才我们是以表[1 2]的两个元素作为最前面的两项来产生费氏数列的，假如让其它任意两个数作为数列的输入元素，它所产生的数列是否也存在着上面的特性呢？

还是让我们继续做一些试验来说明吧。

先将过程 SHU 中的 MAKE "S [1 2]改成：

MAKE "S [-4 3]

然后，输入如下"SHU 13"命令，产生一个 15 项的数列：

-4 3 -1 2 1 3 4 7 11 18 29 47 123 199

试将数列中的最后一项除倒数第二项，键入：

PR 123/199

显示：

.61809

结果仍然十分近似于黄金分割比例值。

费氏数列是一个神奇的数列，除了上面讲的这些特性外，它还有许多秘密。年轻的朋友，开动你的脑筋，去探索、去发掘吧！

第十一章　结构化程序设计

程序设计是计算机开发应用中的一个重要环节。广义上讲，从系统的总体结构设计至程序编制完成，都属于程序设计阶段。

为了顺利地进行程序设计，首先要掌握计算机程序设计语言。但掌握了计算机语言并不等于会编写程序，这正像学习语言和写文章一样，懂得了语言不一定能写出好的文章。因此，掌握了计算机语言仅仅是具备了编写程序的手段，还必须在此基础上继续学习和实践，逐步掌握应用程序的设计步骤和方法。

第一节　结构化程序设计方法

LOGO 语言是一种面向初学者的高级语言，它是由过程组合而成的。过程是相对独立的命令组，可以单独设计、调试和运行。

编写 LOGO 语言程序就像搭积木，将各种功能的过程模块有机的组合在一起，就构成了程序。这种程序结构称为模块化结构。模块化结构程序具有层次清楚、可靠性高、便于阅读和验证、方便递归调用等优点。因此，它在计算机程序设计中得到了广泛的应用。

我们之所以特别强调结构化程序设计，是为了使读者对 LOGO 语言编程树立一个总体的指导思想，养成良好的编程

习惯，为以后学习其它计算机语言打下良好的基础。

一、模块化程序设计原则

结构化程序的设计原则可以概括为两句话："自上而下地设计，自下而上地实现"。

1. 自上而下地设计

自上而下也叫自顶而下。

当我们碰到复杂问题时，首先考虑的不应该是它的各个细节，也不应该全面铺开，而应该从总体着手，把较复杂的问题有机地化成几个子问题。

划分问题的原则是：每一个子问题是原来总问题的一部分，要比总问题容易解决。就是说把顶层（或者说最上层）问题化为第二层子问题。如果第二层子问题还无法全部得到解决，还可以把其中无法直接求解的子问题继续细分成第三层的子问题，依次类推，一层层分下去，直到每一个子问题都能直接求解。这种从总体到部分，步步求精地考虑问题的方法，就是"自上而下"的方法。

2. 自下而上地实现

问题分层完毕，就可以着手解决一个个的子问题，即从最底层开始。

最底层问题最简单，也最容易解决。下层问题解决了，它的上一层问题就得到解决。如此向上一层层递推，使整个问题最终得到解决，这就是"自下而上地实现"。

在 LOGO 程序设计中，上下层次的划分是以主过程、子过程、子过程的子过程……来实现的。从结构上看是主过程嵌套调用子过程，子过程再嵌套调用它的子过程。

二、模块化程序结构的优点

模块化程序结构的优点可以总结为如下几点：

第一、在设计程序时，由于每一步都只细化一个或几个当前优先解决的问题，把精力放在主要问题上，这样就可以以纲带目，使程序设计变得容易。在编写调用过程时，当子过程调试完成后，子过程的正确性就得到保证，上一层过程就有了坚实的基础，问题局部化了，编写和调试程序也变得简单容易。

第二、每一步都有细微的变化，但一环扣一环，步步深入，如果前一步正确就可以保证后一步正确。验证程序的过程较简单。

第三、分层设计，层次分明，使程序易读、易理解、易检查，使算法易交流。在一般情况下，可以不必使用传统的流程图，而用简单的程序树就可以将整个程序描述得清清楚楚。

第四、这种程序设计方法可以培养良好的思考问题和解决问题的习惯。这种方法在数学问题的求解和科学项目的设计中都有很大的用途。

第五、自上而下地设计程序，不仅是LOGO语言的程序设计方法，也是其它电脑程序设计语言的常用方法，如Pascal，BASIC，C语言等。

在使用自上而下的方法设计程序时，如果上层问题和下层问题是类似的问题，只是数据变小或变简单，那么，就可以用递归的方法设计程序，程序会变得非常简洁。

第二节　程序设计标准

一个程序经过运行、调试，得出正确的结果，只表明此程

序可行或无误，但并不一定是最佳程序。求解同一个问题可以用不同的方法，写出不同的程序。其间必有优劣之分。

如何评价一个程序的质量呢？

时代不同、电脑的档次不同，评价优劣的观点、标准也不同。

五、六十年代评价一个程序，往往以为程序运行时花费的机时少、占内存量少就是好程序。随着计算机技术的发展，计算机的运行速度迅速提高，内存容量越来越大，因此效率问题已不再像过去那样突出。相反，由于程序的规模扩大，程序的可靠性问题日益突出。早期的程序以追求效率为主要目标，往往不注意程序的易读性，程序设计无章可循，有时为了追求效率方面的微小改进，使程序晦涩难懂，增加了程序设计、调试和维护过程中的困难，程序的可靠性差。现代科学技术的发展，要求软件的生产方式从“个体方式”中解放出来，按照“工程”的方法组织软件的生产，也就是说，按照一定的规范、一定的步骤来进行程序设计，而不允许程序设计人员随心所欲地编写程序。今天人们在评价一个程序的优劣时，应考虑如下标准：

1. 程序能否按设计说明书的要求正确运行。
2. 软件生产周期短，花费代价低。
3. 便于维护、修改和扩充。
4. 程序结构简洁、明晰、合理。
5. 程序效率高，运行速度快，占用内存小。

第三节　程序设计举例

一、画草人

绘制图 11－1 所示的草人过程，是一个很简单的绘图程序，但不管简单还是复杂，它的设计步骤都是一样的。现以这个简单的图形程序为例，介绍它的设计步骤。

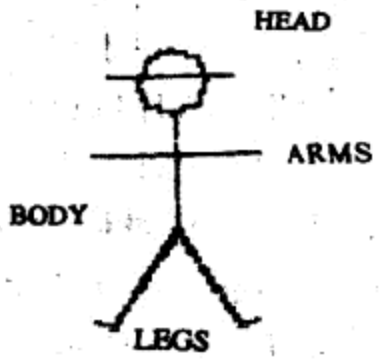


图 11－1　草人草图

1. 程序结构规划

对于画草人的绘图程序设计，可按如下几个步骤来完成：

(1)首先在纸上手工画一个草图(图 11－1)；

(2)把图形分解成若干部分，给每个部分写个名字，这个名字就是相应部件图形的过程名。

(3)画出程序结构图(图 11－2)，表示出过程间的调用关系或各个部分是怎样组合起来的。图 11－2 中，主过程名为“PERSON”、画身体的过程名为“BODY”、“ LEGS”画腿、“ARMS”画臂、“HEAD”画头，“LEFTLEG”、“RIGHTLEG ”分别为画左、右腿的过程，它们为画腿子过程的子过程。

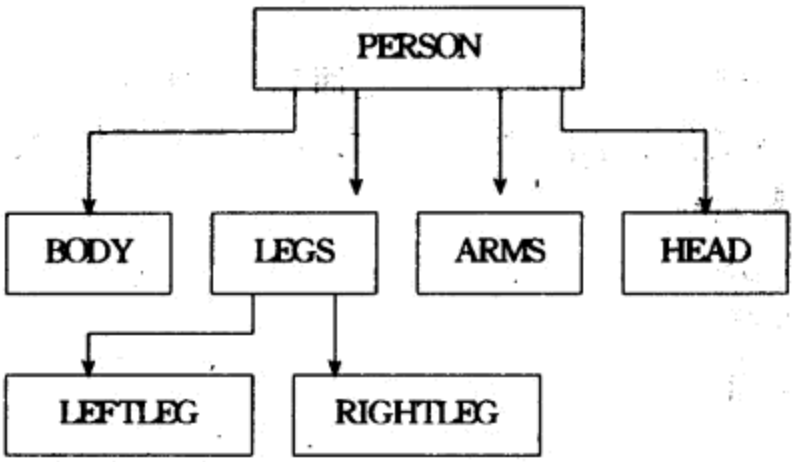


图11－2　程序结构图

2. 编写过程

编写程序，有的人喜欢先编写最小的、局部的过程，再拼出整个图形；有的人喜欢先编写大的过程，最后完善小的过程。前一种方法从结构图最下边编起，自底向上扩展；后一种从结构图最上面开始，自顶向下推进。对于大的程序，以自顶向下的方法为好。我们就以自顶向下的顺序编写这个程序。

(1)主过程

```
TO PERSON
  BODY
  LEGS
  ARMS
  HEAD
END
```

(2)画身体的子过程

```
TO BODY
  FD 30
  BK 30
END
```

这里的 BK 命令是为了使海龟回到出发位置，这会使得以后修改、调试各个过程时比较简单容易，相互牵扯少。

(3)画腿过程

```
TO LEGS
  LEFTLEG
  RIGHTLEG
END
```

① 画左腿

```
TO LEFTLEG
  RT 30 BK 30
  LT 90 FD 5
  BK 5 RT 90
  FD 30 LT 30
END
```

② 画右腿

```
TO RIGHTLEG
  LT 30 BK 30
  RT 90 FD 5
  BK 5 LT 90
  FD 30 RT 30
END
```

(4)画臂

```
TO ARMS
  FD 20 RT 90
  FD 20 BK 40
  FD 20 LT 90
  BK 20
```

```
END
```

(5)画头

```
TO HEAD
  FD 30 LT 90
  REPEAT 36 [RT 5 FD 62.8/36 RT 5]
  RT 90 PU FD 10 LT 90
  PD FD 15 BK 30 PU
  HOME
END
```

上述过程全部定义完后,执行命令

```
PERSON
```

就可画出图 11－3 的图形。

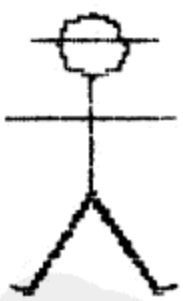


图 11－3　程序绘制的草人

在编写每个过程时,我们都使海龟回到出发位置和状态,这使得程序连接和修改都很容易。现在如果欲绘制图 11－4 所示的图形,可把画胳臂的过程改写如下:

```
TO ARMS
  FD 20 RT 45
  FD 20 BK 20
  RT 90 FD 20
```

```
    BK 20 LT 45
    BK 20
END
```

图 11－4

二、画卡车

我们依照编写画草人程序的步骤，进行画卡车过程的编写。

1. 程序结构规划

(1)手工画出卡车的草图，如图 11－5。

(2)把卡车分解为若干部分，并标上名字。

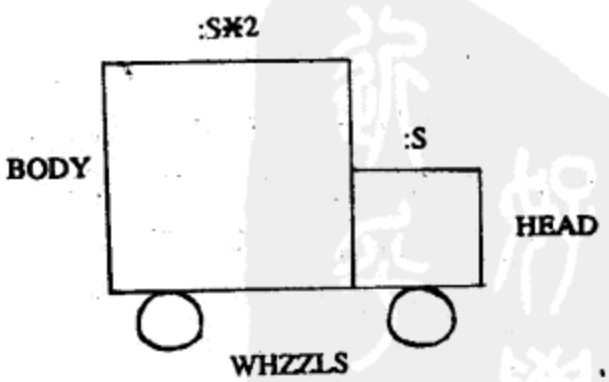


图 11－5　卡车草图

(3)画出程序结构图，写出主过程。程序结构如图 11－6。

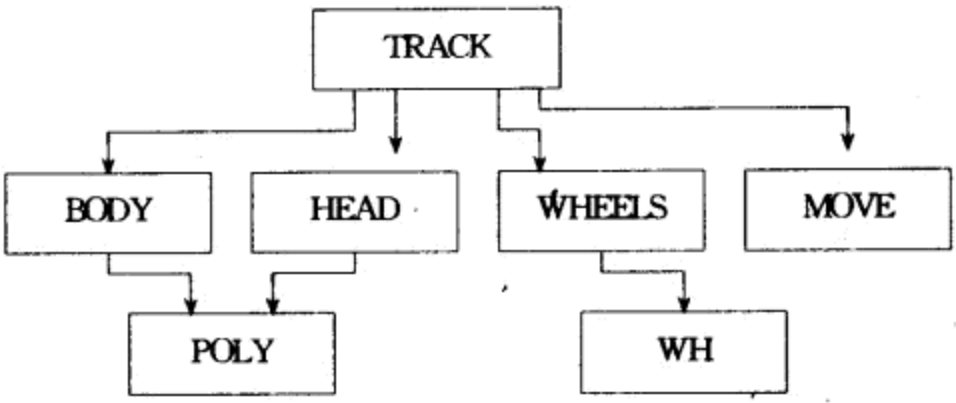


图11－6　程序结构图

2. **编写过程**

(1)编写主过程

```
TO TRUCK S
  BODY 2 * :S
  MOVE 2 * :S
  HEAD :S
  MOVE :S
  WHEELS :S * 2 :S/2
  WHEELS :S * 8 :S/2
END
```

对绘制图 11－5 所示的卡车来说，其实“BODY”和“HEAD”可以使用同一个过程，它们都是正方形，只是大小不同而已。但结构图中分为两个过程，这样编写过程对以后改画其它形状的卡车较方便。

编写具体过程，顺序按结构图，自顶向下、自左向右地编写。

(2)编写画车体的子过程

```
TO BODY S
  POLY :S 4
END
```
(3)编写画车头的子过程
```
TO HEAD S
  POLY :S 4
END
```
(4)编写画车轮的子过程
```
TO WHEELS S R
  SETX XCOR－:S/4
    WH :R
END
```
(5)编写海龟位移的子过程
```
TO MOVE S
  SETX XCOR＋:S
END
```
(6)编写画车厢、车头的子过程
```
TO POLY S N
  REPEAT :N [FD :S RT 360/:N]
END
```
(7)编写画圆的子过程
```
TO WH R
  SETH 90
  REPEAT 36 [RT 5 FD :R * 3.14/36 RT 5]
  SETH 0
END
```
现在已经完成了整个画卡车程序的编写。

以上程序写的较繁琐，留出了较大的优化余地，例如“BODY”和“HEAD”两个过程可以取消（直接使用“POLY”）；画圆过程“WH”可以与画轮子的过程“WHEELS”合并等等。这些仅留给读者练习。

第十二章　应用程序集锦

本章将向读者简单介绍部分应用程序(或称工具程序)和部分复杂图形的程序实例。

第一节　键控绘图程序

LOGO 语言被誉为“电脑教室里的国王”,可以开发儿童的智力,培养孩子们的想象能力。那么几岁的儿童可以开始学习 LOGO 语言呢? 回答是:幼儿园五六岁的小朋友就可以坐在电脑旁做一些适合他们年龄特征的题目,这就需要家长和老师去创造这样的学习环境。

儿童初次接触电脑时,要让他们一下子搞清楚命令、语句等概念将是非常困难的,但若让他们利用键盘上指定的几个键直接指挥海龟做图,却是轻而易举的事情。那么,怎样建立这样的作图环境呢?很简单,只要给几个键赋予新的内容就可以解决。下面先给出定义按键功能的过程清单(过程名为 KH)。

```
TO KH
  ST
  MAKE "X RC
  IF :X="F FD 5
  IF :X="B BK 5
  IF :X="R RT 30
```

```
    IF :X="L LT 30
    IF :X="U PU
    IF :X="D PD
    IF :X="C DRAW
    IF :X="P PICKPEN 0
    IF :X="K PICKPEN 1
    IF :X="S STOP
    KH
END
```

这个过程定义了十个功能键，利用这十个键就可以指挥海龟绘图，每个键就是一条命令，字母键后不必打回车键命令便立即执行。十个字母键所表示的命令如下：

F 海龟前进 5 步。

B 海龟后退 5 步。

R 海龟右转 30 度。

L 海龟左转 30 度。

U 抬笔命令，海龟移动时不画线。

D 落笔命令，海龟移动时画线。

C 清除屏幕图形，海龟恢复初态。

P 删除多余的图形。

K 继续画图。

S 退出运行过程，回到命令状态。

执行过程"KH"就可以使用上述十个键指挥海龟作图，让小朋友们各自发挥自己的聪明才智和想象力。

图 12－1、图 12－2 就是小朋友们用上述所定义的功能键所绘制的作品。

上面我们介绍了定义单个字母功能键的过程，每一个键

图 12—1　家

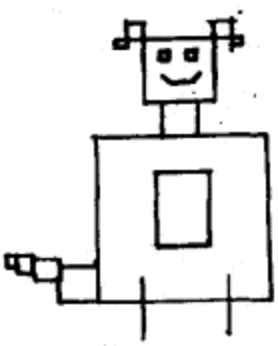


图 12—2　小兵张嘎

所代表的动作，都是我们人为规定的。当然，也可以定义更多的键让海龟去执行更加完善的动作。比如让海龟绕着圆形跑道运动而不出界；画一个小动物，让它运动起来；利用海龟头当作箭头瞄准射击；改变屏幕的颜色等等。

第二节　工具类程序

本节将介绍几个工具类应用程序。

一、画圆及画圆弧的程序

1. 画右弧

```
TO RCP R
  RT 5
  FD :R * 3.14159/18
  RT 5
END
```

2. 画左弧

```
TO LCP R
  LT 5
  FD :R * 3.14159/18
  LT 5
END
```

3. 画右圆

```
TO RCIR R
  REPEAT 36 [RCP :R]
END
```

4. 画左圆

```
TO LCIR R
  REPEAT 36 [LCP :R]
END
```

5. 画左四分之一弧

```
TO LARC R
  REPEAT 9 [LCP :R]
END
```

6. 画右四分之一弧

```
TO RCIR R
  REPEAT 9 [RCP :R]
END
```

R 为圆或圆弧的半径。编写程序中，用到画圆或圆弧时可直接调用或套用以上过程。

二、排序

“排序”是一个非常实用的工具，在各类统计成绩的场合都要用到它。

有一组随机数，

65 54 67 86 69 100 87 77

要求把它们按由大到小的顺序进行排序。

LOGO 语言中虽然没有数组和下标变量，但可用“表”代替数组存放数据。因此，可把排序数据列成表输入电脑。

1. 输入数据的过程

```
TO A
  MAKE "A [65 54 67 86 69 100 87 77]
  OP :A
END
```

2. 寻找最大数的过程

```
TO ZD A N
  IF :A=[ ]
  MAKE "A1 FIRST :A
  IF :A1>:N MAKE "N :A1
  OP ZD BF :A :N
END
```

3. 主过程

```
TO PX A B
  IF :A=[ ] PR :B STOP
  MAKE "A2 ZD :A 0
  MAKE "A3 FIRST :A
  IF :A2=:A3 MAKE "B SE :B :A2
    ELSE MAKE "A SE :A :A3
  PX BF :A :B
END
```

过程“ZD”找出表 A 中的最大元素，放在 N 中输出。主过程“PX”先调用“ZD”过程找出最大元素，放在 A2 中；然后转存到表 B 中，并从表 A 中去掉这个最大元素，此后递归，找出第二个最大元素……。这样每递归一次都找出剩下元素中的最大值，这些最大值依次放入表 B 中，最后表 B 中存放的数据，就是按由大到小的顺序排列的。

第三节　加法练习程序

下面介绍一个两位数加法练习程序。

1. **功能要求**

两位数加法练习程序的基本功能应包括：

· 用户输入每次所练习的题数。

· 电脑随机出题。

· 输入答案、检查答案的正误、累计成绩。如果计算正确，显示“HEN HAO ”(很好)，加一分；如果计算错了则显示“CUO LE”(错了)，给出正确的答案，不计分。

· 检查是否已完成所有的练习题：是则显示得分并停止执行；不是，计数器加 1，继续练习。

2. **编写程序**

下面我们分段编写程序。

```
TO SXT
  CLEARTEXT
  ZS
  MAKE "J 1 MAKE "F 0
  JSQ :J :Z :F
END

TO ZS
  PR [NI XIANG ZUO JI DAO TI?]
  MAKE "Z DU
  END

TO DU
  OP FIRST RQ
END
```

这几个过程的主要作用是询问用户要做几道题，设置题的总数 Z，并设置计数变量 J 的初始值为 1，计分变量 F 的初始值为 0，再告诉计数器 JSQ 作好计数准备。

```
TO JSQ J Z F
  CLEARTEXT
  CHUTI
  HUIDA
  XIA
  IF :J=:Z PR JIESHU TOPLEVEL
  MAKE "J :J+1
```

```
  JSQ :J :Z :F
END

TO CHUTI
  MAKE "SHU1 RANDOM 100
  MAKW "SHU2 RANDOM 100
  MAKE "SHU :SHU1+:SHU2
  PRINT1 :SHU1 PRINT1 [+] PRINT1 :SHU2
PRINT1 [=]
END
```

过程“CHUTI”用来出题，并将正确的结果赋给变量SHU。

```
TO HUIDA
  MAKE "R DU
  TEST PR=:SHU
  IFT PR [HEN.HAO!] MAKE "F :F+1
  IFF PRINT1 [CUO LE,YING GAI SHI:] PR :SHU
END
```

过程“HUIDA”(回答)接受用户输入的数值，由电脑进行分析判断，如果正确，则加分，否则告诉你正确的答案。

```
TO XIA
  PR [QING AN RETURN JIAN]
  MAKE "N RC
END

TO JIESHU
  CLEARTEXT
```

```
    PR SE [NI DE FEN SHU SHI:] :F
    PR SE [GONG YOU] :Z PR [TI]
END
```

过程“XIA”等待用户按回车键，以便出下一道题。最后一个过程“JIESHU”进行结束处理(输出得分)。

3. **使用说明**

执行程序，键入

```
SXT                          (清屏后，显示
NI XIANG ZUO JI DAO TI?      (你想做几道题?
```

键入

```
2                            (回车后，显示
TI 1                         (第 1 题
56+23=79
HEN HAO!                     (很好!
QING AN RETURN JIAN          (请按回车键
TI 2                         (第 2 题
34+43=89
CUO LE,YING GAI SHI 77       (错了，应改为 77
QING AN RETURN JIAN          (请按回车键
NI DE FEN SHU SHI: 1         (你的得分是:1
GONG YOU 2 TI                (共做 2 题
```

这个程序还可作进一步的改进，如：当用户连续答对时，可使题目的难度增加，除加法之外还可以作减法、乘法、除法或简单的四则混合运算练习等等。

第四节　复杂图形绘制程序

本节将奉献给大家三幅生动逼真的图画，供大家欣赏。

一、机器人

过程清单如下：

```
TO JQR
  FULLSCREEN
  A
  B
  C
  D
END

TO A
  PU SETXY(－20) 30
  PD REPEAT 4 [FD 40 RT 90]
  PU SETXY(－15) 55
  PD RCI 5
  PU SETXY(－15) 55
  PD RCI 3
  PU SETXY 5 55
  PD RCI 5
  PU SETXY 5 55
  PD RCI 3
  PU SETXY 10 40
```

```
  PD SETXY 10 45
  SETXY(-10) 45
  SETXY(-10) 40
  SETXY 10 40
END

TO RCI X
  REPEAT 36 [FD 2 * 3.14159 * :X/36 RT 10]
END

TO B
  PU SETXY 0 70
  PD SETXY 10 90
  PU SETXY 6 90
  PD RCI 4
  PU SETXY 0 70
  PD SETXY(-10) 90
  PU SETXY(-14) 90
  PD RCI 4
END

TO C
  PU SETXY(-30)(-35)
  PD SETH 0
  REPEAT 4 [FD 60 RT 90]
  PU SETXY(-10) 25
  PD SETXY(-10) 30
```

```
  SETXY 10 30
  SETXY 10 25
  PU SETXY (-30) (-35)
  PD SETXY (-40) (-30)
  SETXY (-40) 25
  SETXY (-30) 25
  PU SETXY (-43) (-40)
  PD RCI 8
  PU SETXY 30 (-35)
  PD SETXY 40 (-35)
  SETXY 40 25
  SETXY 30 25
  PU SETXY 27 (-40)
  PD RCI 8
  PU SETXY (-20) 10
  PD RCI 5
  PU SETXY (-5) 10
  PD RCI 5
  PU SETXY 10 10
  PD RCI 5
  PU SETXY 15 (-15)
  PD SETH 0
  REPEAT 2 [FD 15 LT 90 FD 30 LT 90]
END

TO D
  PU SETXY (-25) (-35)
```

```
    PD SETXY 0
    REPEAT 2 [RT 90 FD 50 RT 90 FD 60]
    SETXY 0 (-35)
    SETXY 0 (-95)
    PU SETXY (-25) (-85)
    PD SETXY 25 (-85)
    HT
END
```

程序简介：

过程“A”画机器人的脸、过程“B”画机器人头上的天线、过程“C”画机器人的上身、过程“D”画机器人的下身、“JQR”是主过程。

执行“JQR”将画出如图 12－3 所示的机器人。

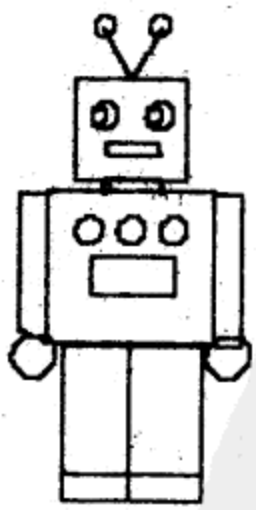


图 12－3　机器人

二、坦克车

坦克车的侧面图由履带、炮台、车顶和五角星标志所组成。下面分别编写绘制各个部分的过程。

1. 画履带和车轮

```
TO T1
  A (-50) (-25)
  RCI 15
  A (-80) (-25)
  RCI 15
  A (-20) (-25)
  RCI 15
  A 10 (-25)
  RCI 15
  A 40 (-25)
  RCI 15
  A 71 (-13)
  RCI 7
  A (-95) (-13)
  RCI 7
  PU SETY (-5) SETX (-90)
  PD SETX 78
  A 85 (-15) B 60 (-40)
  SETX -70 B (-95) (-15)
  PU SETX (-100)
  PD SETH 0 FD 10 B (-90) 0
  SETX 80 B 92 (-5) SETY (-15)
END

TO A
  PU SETXY :X :Y
```

```
END

TO B
  PD SETXY :X :Y
END

TO RCI :Z
  PD REPEAT 36 [FD 2 * 3.14159 * :Z/36 RT 10]
END
```

2. 画炮台

```
TO T2
  A 20 0
  PD FD 5 SETX (-40) BK 5 FD 5
  B (-90) 15 FD 20 B (-20) 40
  SETX 20 B 30 30 SETY 20
  B 25 5 SETX 20
  A 30 30
  PD SETX 110 SETY 25 B 30 20
END
```

3. 画五角星标志

```
TO T3
  A (-15) 35
  PD SETH 163
  REPEAT 5 [FD 25 RT 144]
END
```

4. 画车顶

```
TO T4
  A (-60) 37
  B (-50) 45 B (-30) 50
  SETY 42 SETX (-35) SETY 39
  SETH 0
  FD 6 SETX (-33) BK 3 FD 3 SETX (-30)
  FD 7 SETX (-10) BK 7 SETX (-15) BK 5
  A (-10) 47
  PD SETX 15 FD 4 SETX 10 BK 4 FD 2
  SETX (-10)
END
```

5. 装配

```
TO TANK
  FULLSCREEN
  T1
  T2
  T3
  T4
  HT
END
```

各部件的绘制过程编写完毕，就可以把它们组装起来了。由于在编写各部分的过程时已经考虑到各步的位置，因此在组合时可不必考虑结构的先后次序。你可以将 T1、T2、T3、

T4 这四个过程随便排列，所得出的结果是一样的，这里充分体现了结构化程序设计的优秀特点。

键入“TANK”，屏幕上就会出现一辆如图 12－4 所示的坦克车。

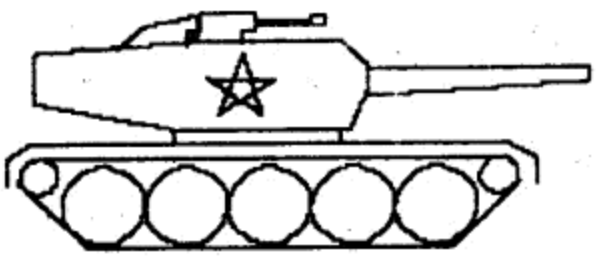


图 12－4　坦克

三、战斗机

以上两个实例采用的是结构化程序结构，下面将给出一个顺序程序结构的示例，供大家参考。

画飞机的过程清单如下：

```
TO FIGHTER
  CLEARTEXT FULLSCREEN HT PD

SETXY "－16 "－20
SETXY "－59 "－58
SETXY "－83 "－76
```

```
SETXY "-70 "-52
SETXY "-29 "-17
HOME
SETXY "-24 "-16
SETXY "-28 "-22
SETXY "-32 "-32
SETXY "-44 "-44
SETXY "-54 "-48
SETXY "-60 "-46
SETXY "-58 "-40
SETXY "-42 "-28
SETXY "-28 "-22
PU SETXY "-58 "-48
PD SETXY "-66 "-54
SETXY "-82 "-76
SETXY "-58 "-64
SETXY "-12 "-26
SETXY "-2 "-10
SETXY "-30 "-16
SETXY "20 "6
```

画飞机头

```
SETXY "83 "-16
SETXY "64 "-32
```

```
SETXY "72 "－26
SETXY "－12 "－26
PU SETXY "0 "－27
PD SETXY "－16 "－38
SETXY "－26 "－38
PU SETXY "－30 "－17
PD SETXY "－44 "22
SETXY "－48 "18
SETXY "－38 "28
SETXY "8 "12
PU SETXY "－42 "－28
PD SETXY "－42 "－20
SETXY "－32 "－12
```

画进气道和前翼

```
PU SETXY "－9 "－4
PD SETXY "－24 "－26
SETXY "32 "24
PU SETXY "26 "5
PD SETXY "40 "44
SETXY "36 "38
SETXY "48 "51
SETXY "46 "24
SETXY "26 "5
PU SETXY "34 "2
PD SETXY "62 "6
```

```
    SETXY "68 "12
    SETXY "50 "18
    SETXY "34 "2
    PU SETXY "50 "18
    PD SETXY "48 "26
    PU SETXY "25 "27
    PD SETXY "24 "32
    SETXY "4 "15
    SETXY "16 "46                    画尾翼、喷射口
    SETXY "13 "43
    SETXY "24 "56
    SETXY "24 "32
    PU SETXY "10 "30
    PD SETXY "2 "32
    SETXY "-2 "28
    SETXY "6 "14
END
```

运行“FIGHTER”将画出如图 12－5 所示的战斗机。

过程“FIGHTER”全部使用 SAETXY 命令编写，可以说毫无技巧可言，但它也表述了一种编程方法。读者可以试用其它程序结构编程，其程序量可以比它还要大。由此说明，在编写绘制不规则图形的程序时，往往最笨的方法就是最合理、最科学的方法。也就是说，采取那种程序结构，要具体问题具体对待，千篇一律必然导致形而上学。

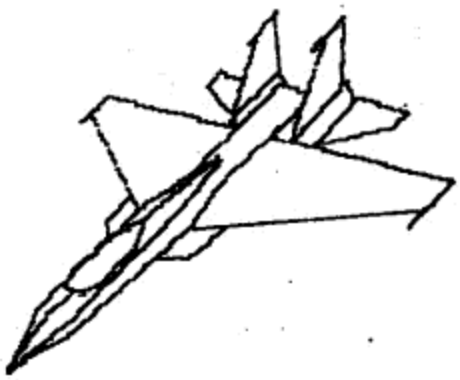


图 12—5　战斗机

至此，海龟奏鸣曲已近尾声，仅愿这个小册子，能伴你进入 LOGO 世界，驾驭海龟在知识的海洋里遨游，去探索无穷的奥秘……。

附录一　错误信息一览表

错 误 信 息	释　　义
1. PROCEDURE NOT FOUND	未发现有此过程
2. VARIABLE NOT FOUND	无此变量
3. MISS)	遗漏右括号
4. FUNCTION NEED MORE INPUT	参数不够
5. FUNCTION NOT LIKE IT AS INPUT	参数错误
6. NO ENOUGH STORAGE	存储空间不够
7. YOU DIDN' T SAY WHAT TO DO WITH	不明错误
8. FUNCTION HAS NO RETURN	函数无返回值
9. FUNCTION CAN ONLY BE USED IN A PROCEDURE	函数只能用于过程中
10. VARIABLE NAME TOO SHORT	变量名过短
11. ERROR IN PROCEDURE DEFINITION	过程定义错误
12. WORD OR LIST EMPTY	字或表已空
13. FLOAT OVERFLOW ERROR	浮点计算数值范围溢出
14. FUNCTION DIDN'T OUTPUT	函数无输出
15. READ ERROR	读磁带出错
16. FD ERROR 1	软盘控制器错误
17. FD ERROR 2	软盘驱动器错误
18. FD ERROR 3	未插软盘
19. FD ERROR 4	软盘格式不对
20. FD ERROR 5	软盘写保护
21. FD ERROR 6	软盘空间不够
22. FD ERROR 7	软盘上没有该文件

附录二　LOGO 命令一览表

一、绘图命令

命令形式	功　　用
BG :X	设置背景颜色，即 0 号笔的颜色，X 的范围是 0—15
BK :N	后退 N 步
BRIGHT :N :X	设定 N 号笔所画颜色的亮度，当 N=0 时设置背景亮度；当 X=0 时为低亮度；X=1 时为高亮度
CLEANTEXT	清除文本屏
CS	清屏，但海龟不回到母位
DRAW	清屏，进入作图，海龟回到母位
FD :N	前进 N 步
FULLSCREEN 或 F3	进入图形屏
GOODBYE	删除一切，返回初始状态
HEADING	输出海龟当前的方向(以度为单位)
HOME	海龟回到母位，但不清屏
HT	隐藏海龟
LT :M	左转 M 度
ND	退出作图，图文全清
PC :N :X	设定 N 号笔的颜色为 X 号代码，X 的范围是 0—15，N 的范围是 1—3
PICKPEN :N	选择 4 支笔(N=0—3)其中 1—3 号笔可以画彩色
PD	落笔画
PU	抬笔不画
REPEAT :N[一组命令]	将中括号内的命令重复执行 N 次
RT :M	右转 M 度
SETXY :X :Y	海龟移到(X,Y)位置
SETX :X	海龟移到横座标为 X 的位置
SETY :Y	海龟移到纵座标为 Y 的位置
SETH :H	海龟转向 H 度位置，尖头向上为 0 度
SPLITSCREEN 或 F2	进入混合屏
ST	显示海龟
TEXTSCRENN 或 F1	进入文本屏
XCOR	输出海龟当前位置的 X 坐标
YCOR	输出海龟当前位置的 Y 坐标
.ASPECT :X	设置横纵比，X 缺省时为 1

二、过程命令

命令形式	功　　用
ALLOF 条件 1 条件 2	两个条件都成立时总条件才成立
ANYOF 条件 1 条件 2	两个条件中有一个成立则总条件成立。
←→↑↓	移动光标
Ctrlk D	删除光标位置的字符
Ctrlk K	删除从光标位置开始直到本语句行末的所有字符
Ctrlk A	光标移到语句行首
Ctrlk E	光标移到语句行末
Ctrlk B	光标上移一屏或移到屏首
Ctrlk F	光标下移一屏或移到屏末
Ctrlk L	将光标当前位置的内容卷至屏首
Ctrlk C	确认定义，退出编辑
Ctrlk G	废除定义，退出编辑
ER 过程名	删除指明的过程
ERALL	删除所有的过程
END	过程结束
Enter(Return)	将光标行的语句分为两个语句行
GO "目标行号：	跳转到目标行号指定的语句行执行。
IF……THEN 语句	如果条件表达式成立，则执行 THEN 后面的语句，否则执行以后的语句
IF……THEN 语句 1	如果条件表达式成立，则执行语句 1，否则执行语句 2
ELSE 语句 2	如果没有 LESE 语句 2，则执行下一语句行
NOT 条件	只有条件不成立时总条件成立
OP	将运算结果返回给调用者
OP：X	中止本过程的执行，将 X 返回给调用者
POTS	显示已定义的几个过成名
PO 过程名	显示指定的过程清单
POALL	显示所有的过程清单
STOP	中止本过程的执行
TO 过程名	定义一个过程
TOPLEVEL	中止过程的执行，回到执行状态
TEST 条件	测试条件是否成立
IFT 执行语句 1	当测试条件成立时执行语句 1
IFF 执行语句 2	当测试条件不成立时执行语句 2

三、运算命令

命令形式	功　　用
ASCII	求字符的 ASCII 码
ATAN :A :B	求反正切值
CHAR :N	求 ASCII 码为 N 的字符
COS :X	求余弦函数值(X 以度为单位)
INTEGER :X	截尾取整数
MAKE "变量 表达式	赋值
ROUND :X	四舍五入取整函数
QUOTIENT :X :Y	X、Y 四舍五入后,对商取整
RANDOM :N	产生 0 至 N 但不包括 N 的随机数
REMAINDER :X :Y	X、Y 四舍五入后,求余数
SIN :X	求正弦函数值(X 以度为单位)
SQRT :X	求平方根(X>=0)

四、字表处理命令

命令形式	功　　用
BF "字	舍去字的首元素,由余下的字符组成新字
BF[表]	舍去表的首元素
BL "字	舍去字的末元素,由余下的字符组成新字
BL[表]	舍去表的末元素
FIRST "字	取字的首元素组成新字
FIRST [表]	取出表的首元素
FPUT 元素[表]	将元素加入表头
LAST "字	取字的末元素组成新字
LAST[表]	取出表的末元素
LIST 元素 1 元素 2	将两个元素组成一个表
LPUT 元素[表]	将元素加入表尾
SE 元素 1 元素 2	将两个元素组成一个表,元素如为表,则自动去掉中括号,它的元素分别成为新表的元素
WORD "字 1 "字 2	把字 1 和字 2 连接起来形成新字

五、输入与输出命令

命令形式	功　　用
CURSOR ,X ,Y	将光标定位在屏幕的(X,Y)处(X=2－20,Y= 3－24)
Ctrl—P	连接打印机打印过程清单
LIST? ,X	检测 X 是不是表
NUMBER? ,X	检测 X 是不是数字
NOTRACE	停止跟踪
OUTPRN	打印绘制的图形
PR 表达式	显示命令
PRINT1 表达式	紧凑显示
RC	等待输入一个字符
READ "文件名	将磁带文件读入内存
READ "A:文件名	将磁盘文件读入内存
RQ	等待输入一行字符
SAVE "文件名	将文件存入磁带
SAVE "A:文件名	将文件存入磁盘
THING "A	输出变量 A 的值
THING? "A	检测 A 是不是变量
TRACE	逐行跟踪显示运行的语句
TS	检测海龟当前的状态
WORD? ,X	检测 X 是不是字

附录三　画笔颜色代码对照表

代　码	颜色	代　码	颜色
0	白	8	黄
1	蓝	9	黄绿
2	蓝紫	10	绿
3	紫	11	深绿
4	紫红	12	浅蓝
5	粉红	13	灰
6	红	14	黑
7	橙	15	透明

附录四　ASCII 码表

ASCII 字符代码表

10 进制代码	16 进制代码	对应的按键	10 进制代码	16 进制代码	对应的符号
0	00		20	14	
1	01		21	15	
2	02		22	16	
3	03	Ctrl+c	23	17	
4	04		24	18	
5	05		25	19	
6	06		26	1A	
7	07		27	1B	Esc
8	08		28	1C	→
9	09		29	1D	←
10	0A		30	1E	↑
11	0B		31	1F	↓
12	0C		32	20	空格
13	0D	Enter	33	21	!
14	0E		34	22	"
15	0F		35	23	#
16	10	Ctrl+P	36	24	$
17	11		37	25	%
18	12		38	26	&
19	13		39	27	'

ASCII 字符代码表（续）

10 进制代码	16 进制代码	对应的符号	10 进制代码	16 进制代码	对应的符号
40	28	(	62	3E	>
41	29	)	63	3F	?
42	2A	*	64	40	@
43	2B	+	65	41	A
44	2C	,	66	42	B
45	2D	—	67	43	C
46	2E	.	68	44	D
47	2F	/	69	45	E
48	30	0	70	46	F
49	31	1	71	47	G
50	32	2	72	48	H
51	33	3	73	49	I
52	34	4	74	4A	J
53	35	5	75	4B	K
54	36	6	76	4C	L
55	37	7	77	4D	M
56	38	8	78	4E	N
57	39	9	79	4F	O
58	3A	:	80	50	P
59	3B	;	81	51	Q
60	3C	<	82	52	R
61	3D	=	83	53	S

ASCII 字符代码表（续）

10 进制代码	16 进制代码	对应的按键	10 进制代码	16 进制代码	对应的符号
84	54	T	106	6A	j
85	55	U	107	6B	k
86	56	V	108	6C	l
87	57	W	109	6D	m
88	58	X	110	6E	n
89	59	Y	111	6F	o
90	5A	Z	112	70	p
91	5B	[	113	71	q
92	5C	¥	114	72	r
93	5D	]	115	73	s
94	5E	^	116	74	t
95	5F	—	117	75	u
96	60	`	118	76	v
97	61	a	119	77	w
98	62	b	120	78	x
99	63	c	121	79	y
100	64	d	122	7A	z
101	65	e	123	7B	{
102	66	f	124	7C	√
103	67	g	125	7D	}
104	68	h	126	7E	×
105	69	i	127	7F	÷

电脑学习机上机指导系列丛书

1. 操作入门
2. 文字处理
3. 游戏BASIC语言
4. 中文BASIC语言
5. LOGO语言
6. 汇编语言
7. 电脑作曲
8. 电脑绘图
9. BASIC语言程序集
10. 硬件与维修

ISBN 7-5053-3072-1

9 787505 330726 >

责任编辑：施玉
特约编辑：李
封面设计：闫欢

书号：ISBN 7-5053-3072-1/TP・1072
定价：9.80元